51CTO.com
技术博客Blog

博客月刊

blog.51cto.com

2014年4月
总第 04 期

- Hadoop原理及部署初探
- Linux—图解PXE实现全自动安装系统
- 一个引号导致1个小时网站打不开
- 构建故障分析平台采用python实现抓包分析数据包
- 职场观察：高薪需要什么？

目录

网络技术：

开发技术:

IT 管理：

Hadoop 原理及部署初探

作者：zuzhou **来源：**http://yijiu.blog.51cto.com/433846/1369225

Hadoop 为何物：

Hadoop 是一个分布式系统基础架构，由 Apache 基金会所开发。

用户可以在不了解分布式底层细节的情况下，开发分布式程序。充分利用集群的威力高速运算和存储。

Hadoop 实现了一个分布式文件系统（HadoopDistributed File System），简称 HDFS。HDFS 有高容错性的特点，并且设计用来部署在低廉的(low-cost)硬件上；而且它提供高传输率(high throughput) 来访问应用程序的数据，适合那些有着超大数据集（large data set）的应用程序。HDFS 放宽了（relax）

POSIX 的要求，可以流的形式访问（streaming access）文件系统中的数据。

Hadoop 基本概念

·核心组件：

MapReduce

HDFS 分布式文件存储系统

·GFS：google file system

将大文件分割成列存储在不同的服务器上

架构与其他分布式文件系统没有太大的区别

·数据类型：

结构化数据 RDBMS

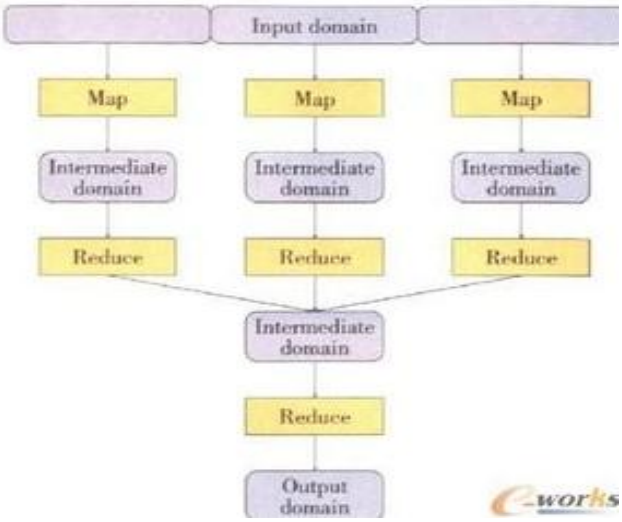
半结构化数据 XML,JSON（轻量级 XML）

非结构化数据

比如：我们使用爬虫对某个站点爬站，最后爬来的数据都是非结构化的数据

再比如：

将一个文件内计算文件中每个单词出现的数量并按照逆序排序--可以将将这个文件按大小分到不同的节点上，每个节点以不额定的文件大小对其进行处理



使用 GFS 时限分部署存储再通过 mapReduce 在分布式存储上完成分布式计算

NDFS:
引入了 mapReduce 思想，于是就有了 HDFS，那么什么又是 MapRdeuce 思想呢？如下

mapReduce 思想：
·编程框架 API 基于这种 API 写的程序能够基于这种模型
·mapredeuce 运行环境 -- Runtime Environment
·mapredeuce implamentation 技术实现

MapRdeuce 基本工作流程：
首先，将单词 MapReduce 拆分为 Map，Rdeuce，顾名思义，此为 2 个组件。
以 google 爬虫为例，在处理的数据只能是键值（key/value）数据，很显然，google 的网络爬虫所爬到的页面，对页面进行 hash 计算，对页面本身当中 key ，对页面的内容当做 value
如果处理的文件为日志，首先日志不是键值，但是 mapReduce 只能处理键值数据，所以这是其 map redeuce 的两段的意义了：
map：将原始数据转换成键值对，将转换后的数据存储下来（过程是按节点来处理的）
redeuce：将处理完成后半段，更进一步处理，完成我们所期望的结果

MapRedeuce 为两段式任务：
（1）所有节点 map 必须都完成处理才可以进入下一段任务
（2）Map 处理阶段至某个进度的时候（这个进度合适度需要自行进行定义），则开启 Redeuce 进程，从而 Map 处理结束后，则进入 Redeuce 处理。
ETL：
抽取转换加载进 hadoop 的工具

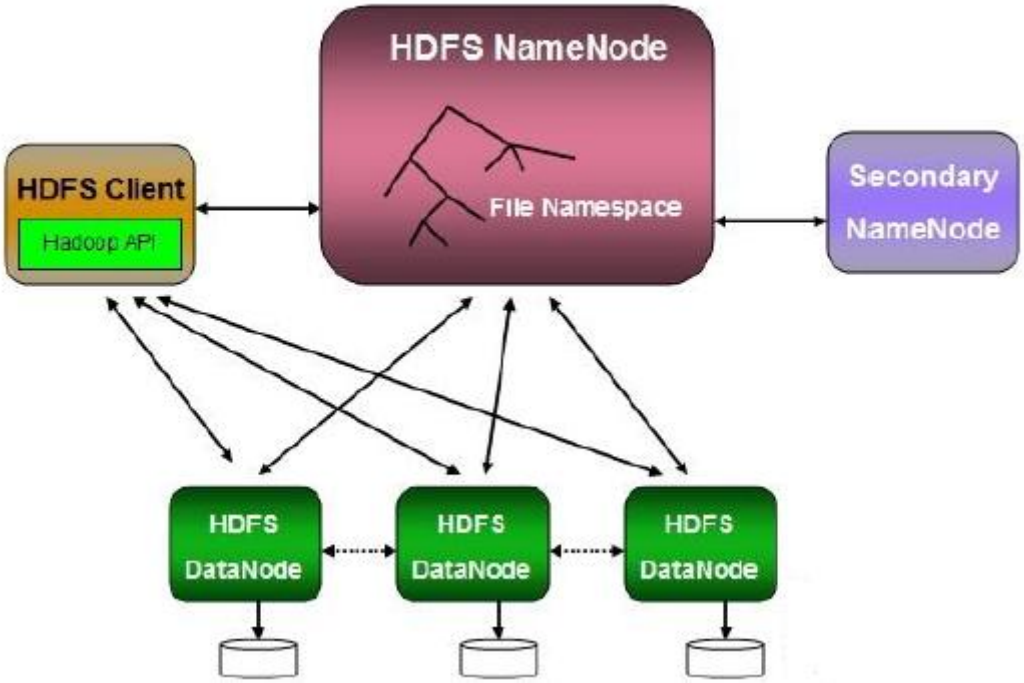
常用 ETL 工具

Hive：实现让 hadoop 拥有 sql 接口，再也不用写任务，但是 Hive 不是实时的的因为 mapReduce 不是实时的
HDFS:工作在用户空间，所有的结合是基于 API 接口，而且只支持写入删除等简单操作，源数据在源数据服务器上存放，数据则在数据服务器上存储，由此如果想在其存放数据并修改的话理论上来说是不可能的
BigTable：工作在 GFS 上，将存储内容最终转换为 HDFS 文件，但是在 bigtable 上支持随机读写，而且是随机高性能读写--列式存储 NoSQL，同一个数据可以存放为 N 个版本
而且可以小量数据读写，但是本身功能依赖于 GFS
impala:为 hadoop 提供实时的接口，使得任务可以实时进行，实时出结果
pig:yahoo

这样的工具很显然不能满足我们目前的需求，于是，HBase 诞生了：
HBase：工作在 HDFS 之上，利用 HDFS 的功能实现了 NoSQL 列式存储的方式，而其就是 Bigtable 的克隆版，因此可以在 HBase 基础上也可以随意修改数据了；
HBase 本身有分布式功能 ,因此跑在 HDFS 天然具有分布优势 ,而 HBase 自己能够实现自动分区的功能 与 mogodb 相似，由于是列式存储，所以性能非常好。
因此做日志分析或日志挖掘，基于 HBase 就可以完全实现

MapReduce 工作在 HBase 上如果不基于 MapReduce 而是跑在 HDFS 是没有问题的，但是只能查看结果而不能处理结果

Hadoop 组成部分：
hadoop 由 JAVA 语言所研发，其不适合存储海量零碎小文件，其关键组件有：

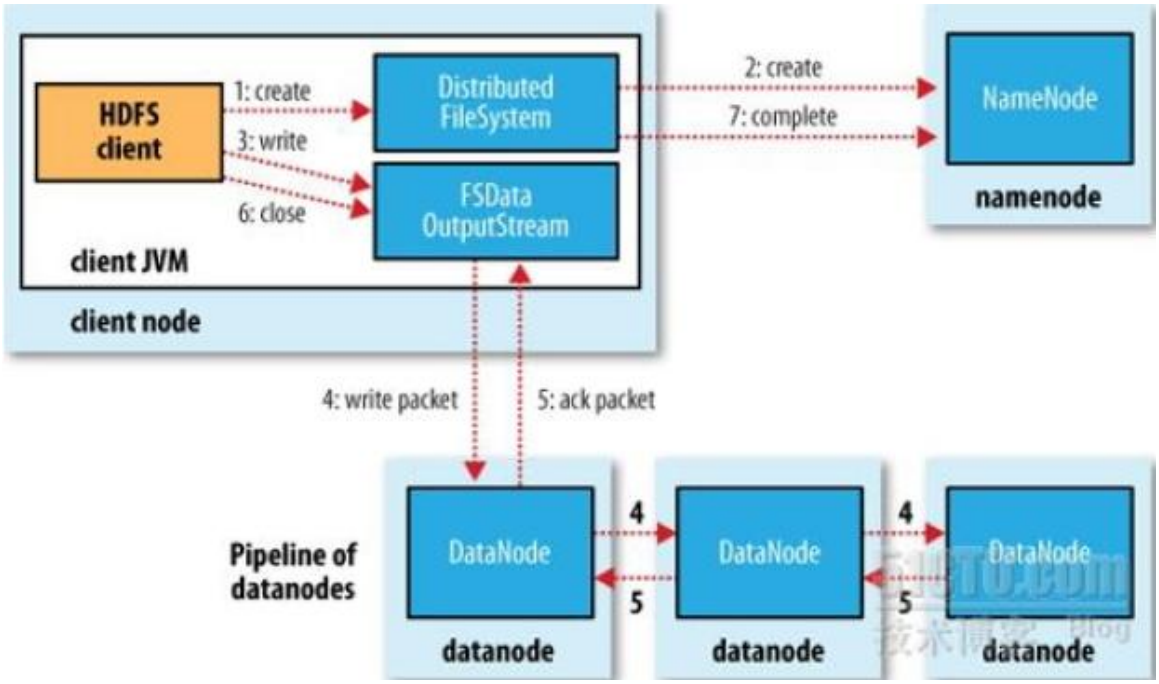


·NameNode：名称节点，主要功能在于实现保存文件元数据，这些元数据直接保存在内存中，为了保证元数据的持久性，而也会周期性的同步到磁盘上去。磁盘上的数据通常被称为元数据的映像数据 image file 以防万一 如果服务或机器崩溃了 ,它会基于Image File 以及各个 DateNode 的报告信息重新生成元数据。

·Secondary NameNode：第二名称节点，NameNode 在早期只有一个，后来提供了第二个名称节点 Secondry NameNode ,万一主节点崩溃，secondary 无非是将 image file 整合到本地实现快速启动节点而已，同时也能够对 Namenode 节点映像文件合并的功能，平时不提供任何节点的服务。

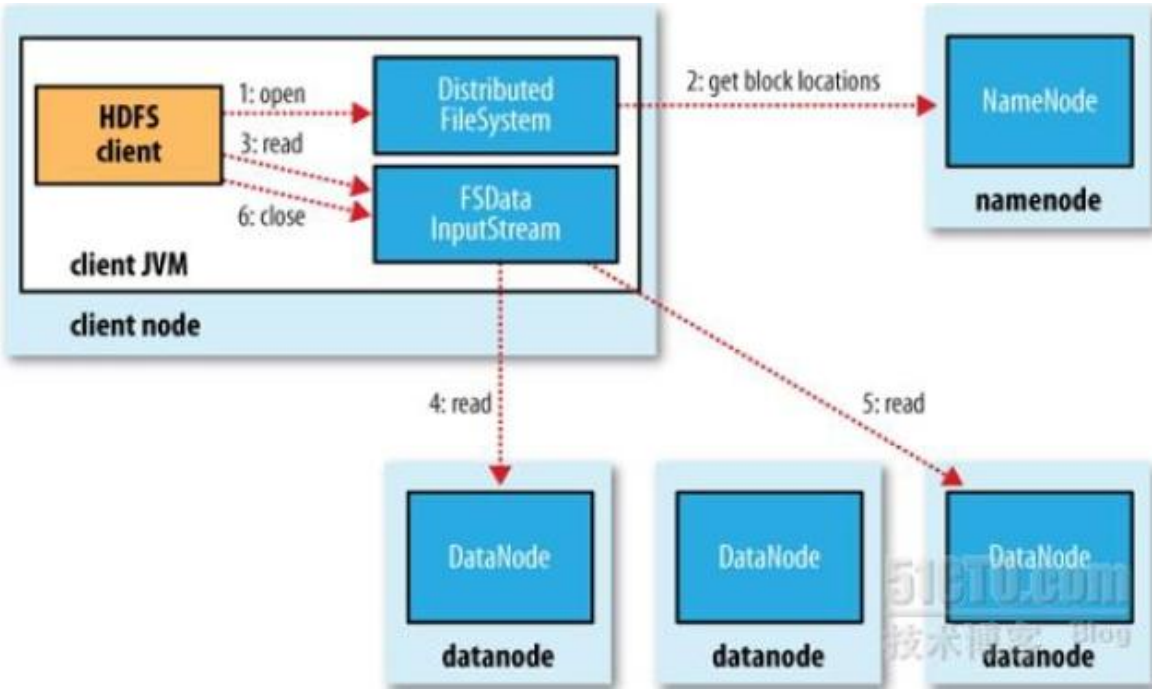
·HDFS DataNode：所有大数据都保存在数据节点上，我们称为 DataNode

服务是如何在 HDFS 中写数据



（1）当我们创建一个数据时候，需要先向 NameNode 发起请求；
（2）NameNode 收到请求之后，会告知 HDFSDataNode，说明需要分别存储至什么位置，当报告返回回来之后，HDFS Client 将对其发起写操作请求；
（3）NameNode 会为 HDFS Client 会分配一个 DataNode 数据节点；
（4）数据节点可能是多个，但 HDFS Client 却无知这一点，只需要向其一节点发起写操作即可；
（5）第一个数据节点 datanode 接到请求之后，自行将数据复制到其它节点
（#一个数据块 报文为 64k 于是其按报文逐一开始存放）
（为了保证数据可用性以及降低集群的成本，hadoop 是工作在商用计算机硬件（说白了就是服务器或专业级存储），而且服务器不需要做 RAID,它会在磁盘上自行存储多个副本，而且是在不同主机上）
（6）每个数据块存储结束之后，数据节点 DataNode 都要向 Namenode 报告存储完毕，接着存放下一个数据块然后报告。。。以此类推；
（7）一旦所有数据都存储完成，NameNode 会保存一个列表，记录着数据的副本保存在哪个数据节点上。

如何在 HDFS 中读取数据



比如文件 file1 分别存储在 D1 D2 D3 上，所以一旦有请求读取数据时，那么 3 个节点都有数据，那么该找谁去读取呢？如上图所示：

当客户端要请求访问某个数据块的时候，一个数据块可能在多个节点都有，那么所以这时候名称节点 NameNode 会告知它第一个块或文件 分别存储在 D1 D2 D3 这么 3 个节点上，那么我们的客户端将会去第一个节点 datanode1 上去取数据，datanode1 接到请求，将数据返回给客户端，假如中途出现中断的情况，那么客户端则去找 datanode2，datanode2 接到请求后如果有数据那么则将数据返回给用户
其实用户请求的为列表，告知的节点上都存在同样的数据；

如果用户请求存放某大文件，那么会被分割为报文，以报文形式存储到 datanode 再由 datanode 相互复制 从而达到并行存储；

同样，也可以并行去读取文件，如果某台数据节点出现问题，那么肯定会导致数据不统一，所以每个 datanode 每隔 3 秒钟会向 namenode 报告自己的心跳信息、所持有的数据块的列表，如果超过阀值则将移除可用列表

在向 namenode 报告时候为了避免报告出错，还要检验一次数据块是否有问题，如果没有问题则报告，如果 namenode 没有接到其报告信息，则认为这个数据块缺少副本，并找将报告的完整副本完全复制到其故障节点上

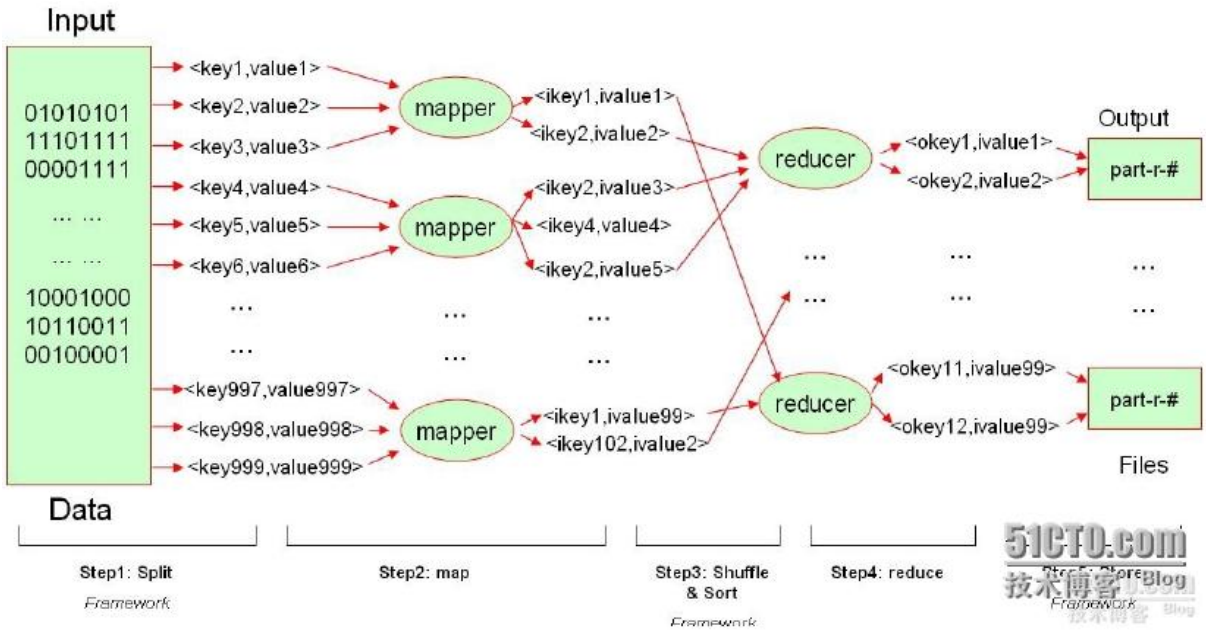
Hadoop 名称节的可用性

最简单的方式将名称节点上的持久元数据信息存储多个副本于不同的存储设备（独立硬盘，NFS 等网络存储文件系统）中

·第二名节点
Secondary NameNode
·负载均衡

HDFS 要做初始化硬盘
·支持回收站功能
·有专用自己的客户端工具
Map Reduce
·函数式编程框架 类似于能够接受参数、传递参数、处理机制等并将 Map Reduce 映射到集群上
大致为：
某一个数据存放在集群内多个节点上，有一数据在其中 A、C、F 三个节点上。
通过 namenode 可以得到其数据有多大，以及在哪些节点上，于是派发任务至存放期数据的节点上，做”Map”处理：
Map 将这些文件，切割成一个一个的键值对，每个键值对应一个数值，而后输出由 Reduce 处理

JobTracker -- 作业追踪器
继续上面，由 MapReduce 集群的 JobTracker 决定来启动多少个 Map 作业，由此，在 A C F 节点上分别启动，但是对于 mapReduce 节点来讲是公共的，如下图所示:



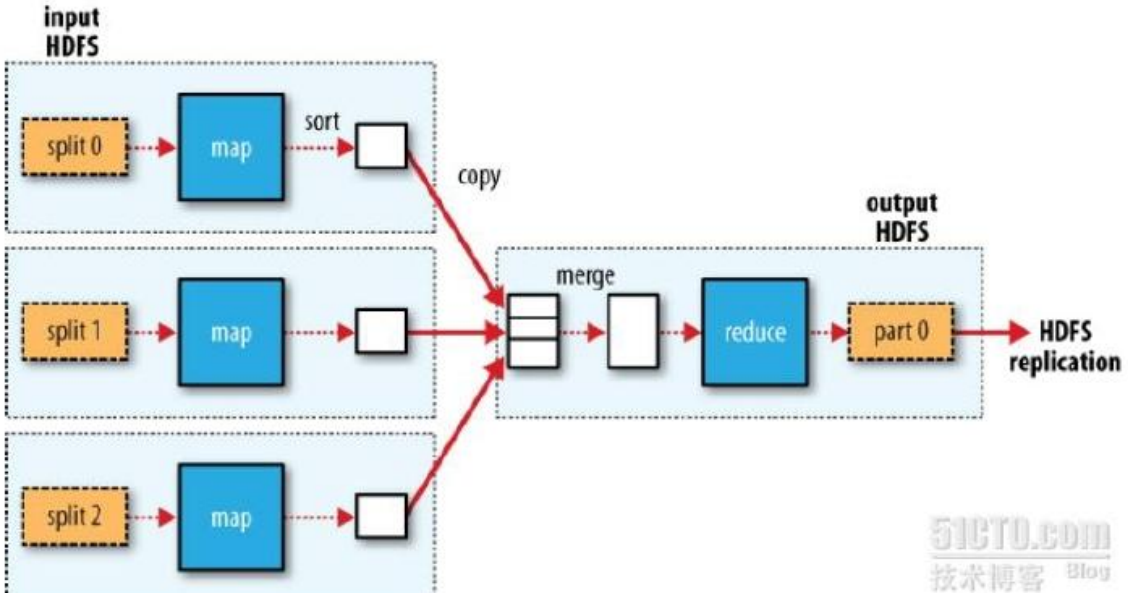
(DataNode 也被称为 TaskTracker；
NameNode 也被称为 JobTracker)

tasktracker 接收到 jobtracker 任务之后，就在本地节点（datanode）启动一个 map 作业（进程），说白就是启动一个 java 虚拟机。所以如果运行 3 个作业 分别在 A C F 节点上也就意味着各自提供一个任务进程，这 3 个任务进程实现去读取文件并完成进一步处理;
在实现 map 之前还要执行 split 操作：

将整个文件完全提取成键值对，这些键值对会均衡发往 maper 任务进程，每个 maper 拿到作业进程之后会对键值对在本地进行处理，运行之后会将 split 分割好的并分派给 mapper 的任务做输出，所以输出定是键值对，但未必和原有的一样;

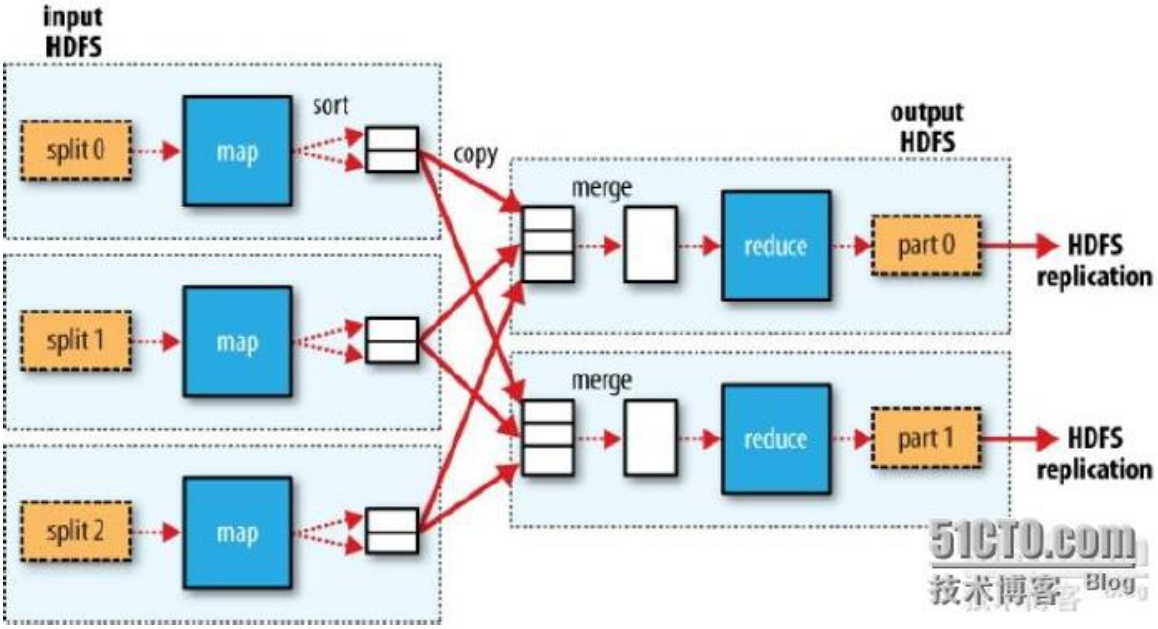
于是这些键值要发往多个 Reducer ，Reducer 也是由 jobtracker 发起的,一旦 maper 快要结束，那么必须要启动 Reducer 来接受 3 个 map 进程本身处理结果的 ,主要是保证同一个键值发往指定的 Reducer。总而言之，只要键值相同，那么肯定是发往同一个 Redeucer

单 redeuce 任务的 MapReduce 数据流



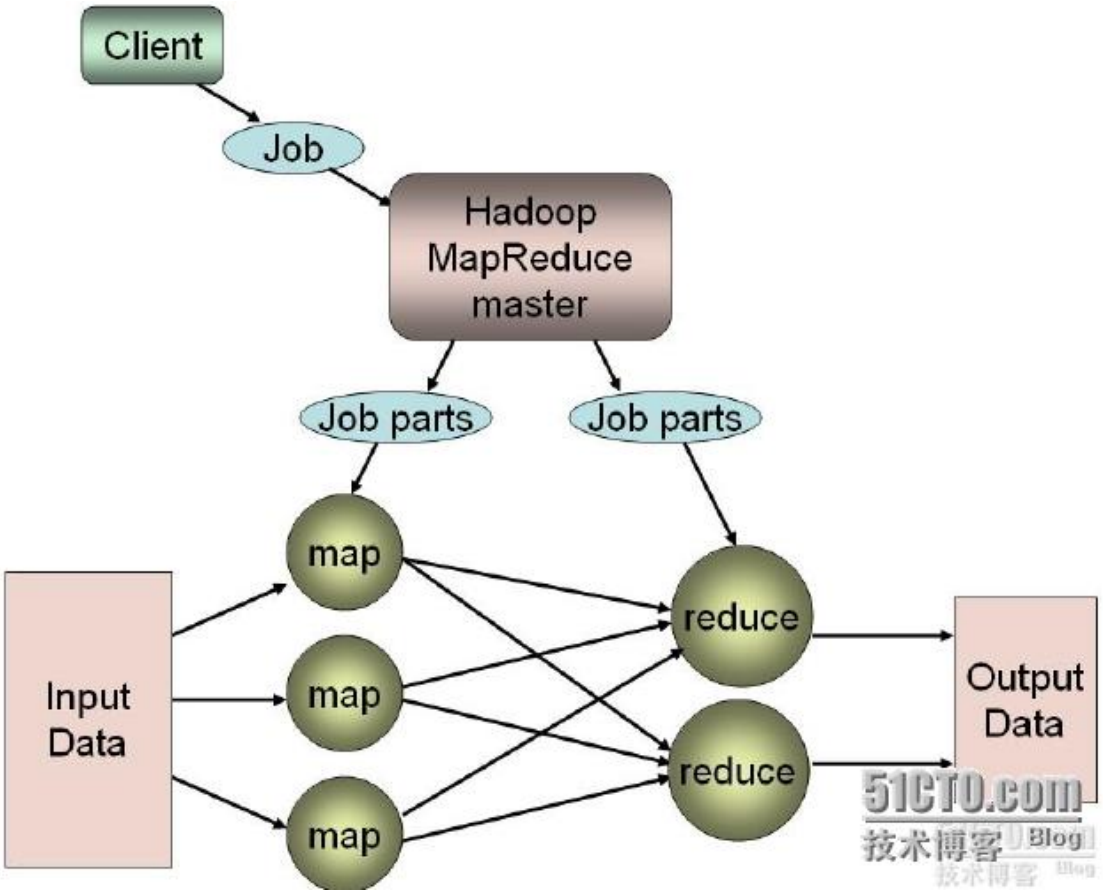
将大文件切割成 N 份，并平均分配多个 soplit 任务 每个 map 处理完之后要在本地排序处理，主要目的是将每个键值对 发给同一个 Reduce 而我们只有一个 Reduce ,Reduce 在处理之前由 merge 将同键值对合并处理并交给 Reduce；Reduce 将结果输出并保存在 HDFS 上保存为副本文件

多 redeuce 任务的 MapReduce 数据流



一个作业可以没有 redeuce 但绝对不能没有 map

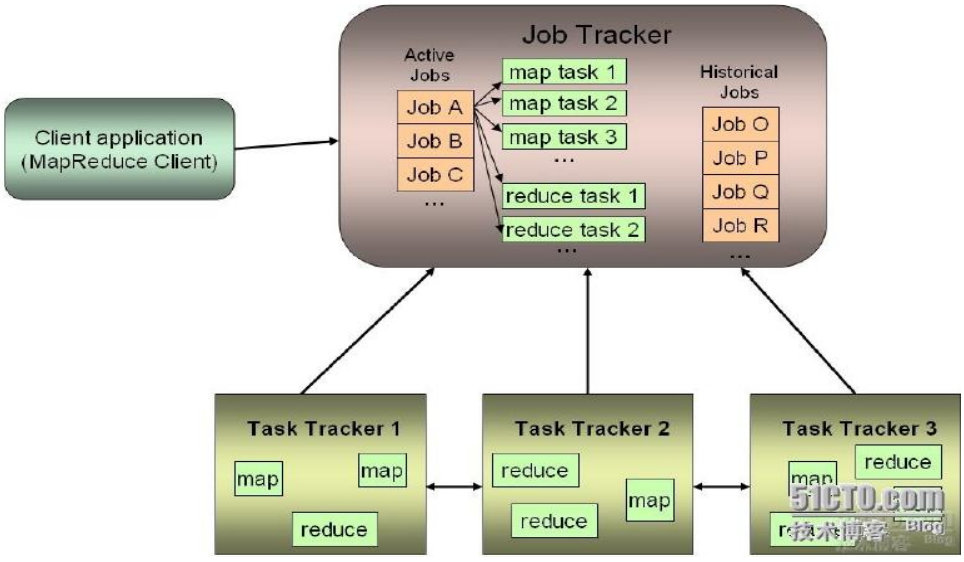
客户端如何提交作业的



job 将作业发送给 master （jobtracter）其控制着 3 个作业（Map）;

每个 Map 去分割完数据之后并在本地完成处理，并且将处理结果按机制向对应的 Reduce 发送数据；最终 master 要在 Map 处理完成之前启动多个 Reduce 并接收来自 maper 的数据，并完成本地数据库处理而后输出结果；

MapRedeuce 逻辑架构



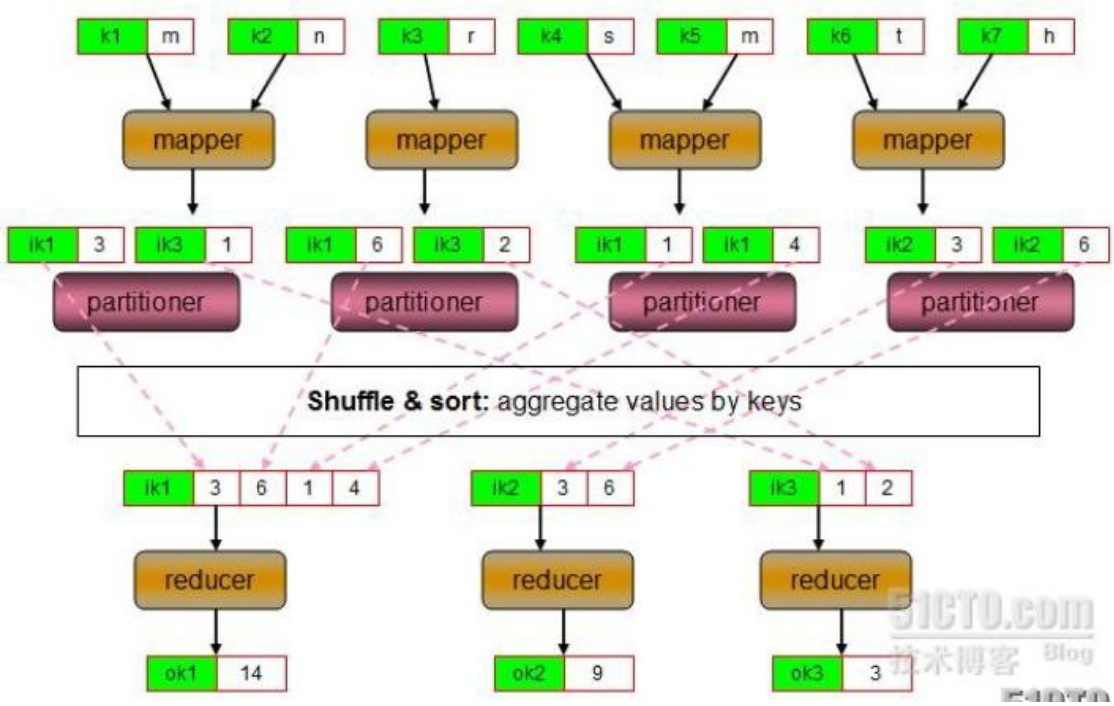
一个 hadoop 集群，其中节点可能是成千上万的，同时运行的作业未必有一个 ，提交的 mapperredeuce 可能会同时运行 但是有个基本前提就是一般 task tracker 节点上最多有 2 个 map 和 2 个 reduce 服务，由此 top tracker 必须自行定义这些作业到底怎么跑跑在哪个节点上。

Historical Jbos 表示已经跑完的作业。

Active 为正在跑的作业。如上图可以看到 task tracker 分别跑的 reduce 数量不一，是由 job tracker 根据数据摆放的机制调度来决定的

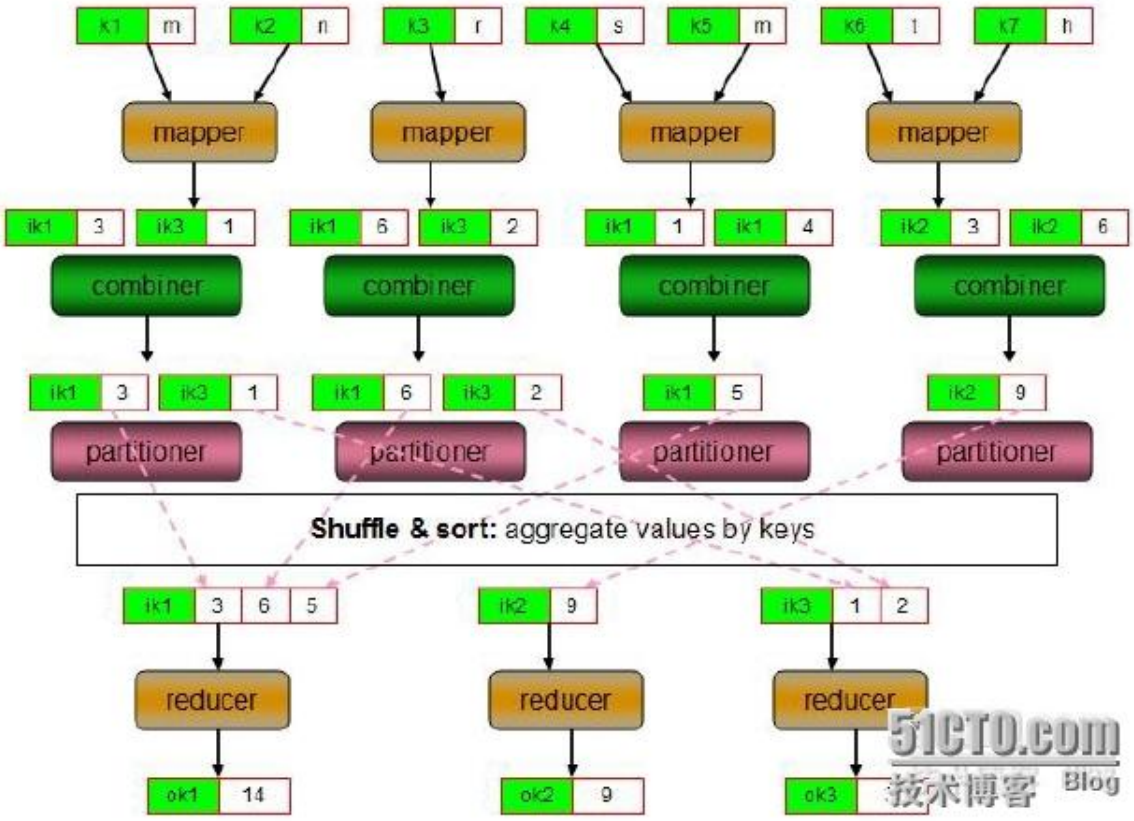
Partitioner 和 Combiner

Partitioner



Partitioner 主要作用是将 mapper 的结果排序完成并将结果输出结果给定义的 reduce 所以每个 mapperd 背后对应一个 partitioner 程序，这个程序用户实现将本地的键输出的结果排序并根据整体 key 的数据发送给 reducer

Combiner



Combiner :在必要时,如果需要最后由 redeuce 进行合并那么 Combiner 先在本地进行合并,Combiner 不能在本地修改键

Hive 和 Pig

能够将提交的任务提交给 jobtracker，由其转义之后才可以运行

·Hive

在目前来讲被理解为数据仓库架构

能够帮助用户将数据存储至 HDFS

需要存放数据则与 HIVE 交互 如果需处理数据则需与 mapreduce 交互

功能：ETL：能够实现数据的抽取和加载，实现了数据存储管理，能够对大型数据集的查询和分析，能够跟框架进行交互，但也有自己的工作框架：

·Pig

与 Hive 一样，而 pig 本身是一个编程平台，内置转换编译器，也可以运行单节点或分布式环境。能够利用 hadoop 大规模数据分析的脚本语言和运行框架

HDFS

HBase 功能

·实时访问

·随机访问

其本身是一个开源的分布式、支持多版本的面向列式的存储系统，实际上 HBase 存储的依然是键值对。

为了能够使 HBase 存储海量数据，建议将 HBase 跑在 HDFS 上，这样效果会更好，但无论如何，如果 HBase 想要实现自动分区等高级操作，节点可能会不止一个。
要结合 Zookeeper 才可以实现工作，所以，要想实现分布式存储必须先实现 zookeeper

HBase 将经常放在一起访问的数据排版为单独一个字段，这样的机制被称为列族

列族：
列族内的数据是放在一起的，所以往表内插入数据，可以只往期字段上插入数据其他字段可以同时没有数据

HBase 特性：
·线性模块可扩展性
·一致性读写
·可配置表的自动切分策略
·RegionServer 自动故障恢复
·基于 MapRdeuce 备份
·提供便利的 API
·为实时查询提供块缓存

Zookeeper
·专门设计为分布式应用所设计的开源、协调的服务，可以为用户提供同步、配置管理、分组和名称空间管理功能
·支持 java 和 C 两种接口
·zookpeer 最少需要 3 个节点，可以多，但不能少，尽可能是奇数节点；
·zookpeer 也是文件系统，但是文件系统是虚拟出来的，在内存中模拟出来的，被称为名称空间
名称空间：内部的节点叫 znode
其名称空间分为 2 类：
·永久节点
·临时节点
Impala:实时分析和处理

部署 hadoop
1.搭建伪分布式环境
1.1 规划如下

IP 地址	服务角色	操作系统
172.23.214.50	ALL	Debian 7.2

1.2 准备工作

配置本地 hosts，将本地 ip 地址解析至 localhost，至于为什么，往下就会明白

```
root@localhost:~# cat /etc/hosts
127.0.0.1       localhost
root@localhost:~# hostname
localhost
```

创建普通用户并授权 ssh key 免密码登录

```
root@localhost:~# useradd hduser
root@localhost:~# passwd hduser
```

授权本地 ssh key

```
hduser@localhost:~$ ssh-keygen -t rsa -P ''
hduser@localhost:~$  cd
hduser@localhost:~$ ssh-copy-id -i .ssh/id_rsa.pubhduser@localhost
```

测试本地 ssh key 是否生效

这里直接 ssh 到目标主机，并执行 date 命令 查看是否生效

```
hduser@localhost:~$ ssh hduser@localhost ; date
Wed Mar  5 21:47:40 CST2014
```

安装 JDK 1.7

这里直接将其他主机的 jdk 打包并解压缩到本地

```
hduser@localhost:~$ mkdir -p /home/hduser/jdk
hduser@localhost:~$ tar xf jdk.tar.gz
```

配置 JDK 环境变量

```
root@localhost:~# cat /etc/profile.d/java.sh
JAVA_HOME=/home/hduser/jdk
HADOOP_HOME=/usr/local/hadoop
STORM_HOME=/home/storm
export JAVA_HOME HADOOP_HOME STORM_HOME
PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$STORM_HOME/bin/
```

激活配置文件

```
root@localhost:~# source /etc/profile
```

查看变量是否生效

```
root@localhost:~# echo $JAVA_HOME
/home/hduser/jdk
root@localhost:~# java -version
```

```
java version "1.7.0_03"
OpenJDK Runtime Environment (IcedTea7 2.1.7) (7u3-2.1.7-1)
OpenJDK 64-Bit Server VM (build 22.0-b10, mixed mode)
```

解压 hadoop 安装包

这里用的包装包为 hadoop-2.2.0 64 位,并解压缩至/usr/local/目录下

```
root@localhost:/usr/local#   ln -s /usr/local/hadoop-2.2.0//usr/local/hadoop
```

设置 Hadoop 环境变量

```
root@localhost:cat /etc/profile.d/hadoop.sh
HADOOP_HOME=/usr/local/hadoop
PATH=$HADOOP_HOME/bin:$PATH
export HADOOP_HOME PATH
```

查看是否生效

```
root@localhost: hadoop version
Hadoop 2.2.0
Subversion Unknown -r 1556437
Compiled by hadoop on 2014-01-08T04:16Z
Compiled with protoc 2.5.0
From source with checksum 79e53ce7994d1628b240f09af91e1af4
This command was run
using/usr/local/hadoop-2.2.0/share/hadoop/common/hadoop-common-2.2.0.jar
```

对目录进行授权

```
root@localhost:/usr/local# chown hduser.hduser -R/usr/local/hadoop/
```

配置伪分布式：

配置的比较关键的配置文件为以下几个：

```
hdfs-site.xml
mapred-site.xml
core-site.xml      #对于 hadoop 来讲是根据属性来定义的，而这些属性其实就是 java 运行的属性定义
hadoop_env.sh      #设置 hadoop 环境变量脚本 ,必须要依赖 jdk ,如果没有在全局配置文件里设定的话，可以在这个文件里去设置变量
```

配置 core-site.xml

```
hduser@locahost:/usr/local/hadoop/etc/hadoop$ cat core-site.xml
```

```
<configuration>
 <property>
   <name>hadoop.tmp.dir</name>          #示 hadoop 存放数据的目录 ,即包括 NameNode 的数据，
也包括 DataNode 的数据，路径可以任意，但必须确保路径的存在和路径的属主是否为 hduser
  <value>/hadoop/temp</value>        #如果不指定路径，则默认是/tmp/
 </property>

#定义 HDFS 默认名称节点（主节点）的路径以及监听端口
<property>
  <name>fs.default.name</name>        #定义 HDFS 的名称节点和其默认的文件系统
 <value>hdfs://localhost:8020</value>
</property>
</configuration>
```

切换至管理员创建目录

```
root@localhost:~# mkdir -p /hadoop/temp/
root@localhost:~# chown hduser.hduser -R /hadoop/
```

定义 mapred-site.xml

```
hduser@localhost:/usr/local/hadoop/etc/hadoop$ cpmapred-site.xml.template mapred-site.xml
hduser@localhost:/usr/local/hadoop/etc/hadoop$ catmapred-site.xml
<configuration>
 <property>
    <name>mapred.job.tracker</name>            #接受多个以逗号分隔路径列表作为其值，并会以轮
流的方式将数据分散存储在这些文件系统上
    <value>localhost:8021</value>             #设置 tracker 的默认端口
 </property>

 <property>
      <name>mapred.map.tasks</name>            #具体到底产生多少个分片进行处理，因为多少个
map 是有关系，这里为 10 片
      <value>10</value>
 </property>
```

```
<property>
    <name>mapred.reduce.tasks</name>          #每任务的 reduce 数量
    <value>2</value>
 </property>
</configuration>
```

定义 HDFS hdfs-site.xml

```
<configuration>
 <property>
    <name>dfs.replication</name>          #定义 dfs 节点数，这里我们只是本机，所以写 1
    <value>1</value>
 </property>
</configuration>
```

配置完毕，接下来初始化名称节点空间

```
hduser@localhost: hadoop namenode -format
```

或者

```
hduser@localhost: hdfs namenode -format
```

启动 hadoop

```
hduser@localhost: pwd
/usr/local/hadoop/sbin
```

如果 job tracker 与 namenode 在同一节点上，那么可以使用一个脚本 start-all.sh 全部启动起来,如果不在同一主机上，则需要逐个启动，以免增加服务器压力

```
#hadoop-daemon.sh       #专门在某个节点上启动某个进程，如果我们新加了一个 datanode ，显然不能将整个集群重新启动这是不现实的，所以要在新加的节点上配置好之后，启动 hadoop-daemon.sh 后面跟上参数 datanode | tasktrack .等 即可
```

我们来启动所有服务进程

```
hduser@localhost:start-all.sh
```

使用 jps 来查看启动的任务进程

```
hduser@localhost:~$ jps
7260 ResourceManager
7720 Jps
7354 NodeManager
6867 NameNode
```

```
6958 DataNode
7124 SecondaryNameNode
```

测试：向 hadoop 中保存文件

```
#在 HDFS 文件系统中新建目录
duser@localhost:/usr/local/hadoop/sbin$ hdfs dfs -mkdir /input
#查看目录
hduser@localhost:/usr/local/hadoop/sbin$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - hduser supergroup          0 2014-03-03 15:02 /input

#上传脚本至于 HDFS 根目录
hduser@localhost:/usr/local/hadoop/sbin$ hdfs dfs -putstop-all.sh /
#查看是否上传成功
hduser@localhost:/usr/local/hadoop/sbin$ hdfs dfs -ls /
Found 2 items
drwxr-xr-x   - hdusersupergroup          0 2014-03-0315:02 /input
-rw-r--r--   1 hduser supergroup       1462 2014-03-03 15:03 /stop-all.sh
```

验证：

运行 Hadoop 自带的 wordcount 程序实现对测试文件中各单词出现次数进行统计的实现过程

```
hduser@localhost: wgethttp://www.gutenberg.org/cache/epub/20417/pg20417.txt
hduser@localhost: hdfs dfs -ls /tmp
hduser@localhost: hadoop -jar
/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jarwordcount   /tmp/ /tmp/
hduser@localhost: hadoop fs -ls /tmp-output/part-r-00000
hduser@localhost: hadoop fs -get /tmp-output/part-r-00000
```

#在执行过程中我们可以看到有大量的信息，如下所示：

```
14/03/04 15:52:09 INFO mapred.LocalJobRunner: wrote record24000. 820459080 bytes left. > map
14/03/04 15:52:12 INFO mapred.LocalJobRunner: wrote record 36800. 685632166bytes left. > map
14/03/04 15:52:15 INFO mapred.LocalJobRunner: wrote record 48600. 561878098bytes left. > map
14/03/04 15:52:18 INFO mapred.LocalJobRunner: wrote record 60000. 442778615bytes left. > map
14/03/04 15:52:21 INFO mapred.LocalJobRunner: wrote record 72400. 312188642bytes left. > map
14/03/04 15:52:24 INFO mapred.LocalJobRunner: wrote record 80400. 227760952bytes left. > map
```

```
14/03/04 15:52:27 INFO mapred.LocalJobRunner: wrote record 94600. 77905011bytes left. > map
14/03/04 15:52:29 INFO mapred.LocalJobRunner: wrote record 94600. 77905011bytes left. > map
```

就是说不管你执行哪个示例程序，启动的 job 都只是在 master 这个节点本地运行的 job，也就意味着为单节点工作

完全分布式

1.规划如下：

IP 地址	服务角色	操作系统
172.23.214.50	Master（NameNode）、JobTracker	Debian GNU/Linux 7.2
172.23.214.47	Slave (Secondary NameNode)	Debian GNU/Linux 7.2
172.23.215.61	DateNode 、Task Tracker	Debian GNU/Linux 7.2

2.准备工作

2.1 修改本地 hostname 主机名，并指定 hosts 配置文件

以 master 为例：

```
hduser@namenode1:~$ hostname
namenode1
hduser@namenode1:~$ cat /etc/hosts
172.23.214.50   namenode1
172.23.214.47   namenode2
172.23.215.61   datanode1


127.0.0.1 localhost
```

测试链路是否通畅

```
hduser@namenode1:~$ ping datanode1
hduser@namenode1:~$ ping namenode2
```

反之，在其他节点也对其主机之外的 host 进行的测试，确保链路畅通

2.2 配置 ssh 免密码登录

首先确保系统是否存在 hduser 或其他自定义用户名，如没有则手动建立

```
root@namenode1:~# id hduser
uid=1002(hduser) gid=1002(hduser) groups=1002(hduser)
root@namenode1:~# su - hduser
```

在各节点主机都进行以下操作

```
hduser@namenode1:~$ cd
hduser@namenode1:~$ ssh-keygen -t rsa -P ''
hduser@namenode1:~$ ssh-copy-id -i .ssh/id_rsa.pub hduser@datanode1
hduser@namenode1:~$ ssh-copy-id -i .ssh/id_rsa.pubhduser@namenode1
hduser@namenode1:~$ ssh-copy-id -i .ssh/id_rsa.pubhduser@datanode2
```

验证免密码

```
hduser@namenode1:~$ ssh hduser@namenode1 date
2014 年 03 月 06 日 星期四 11:21:22 CST
hduser@namenode1:~$ ssh hduser@namenode2 date
2014 年 03 月 06 日 星期四 11:21:24 CST
hduser@namenode1:~$ ssh hduser@datanode1 date
2014 年 03 月 06 日 星期四 11:21:30 CST
```

2.3 环境变量

配置 JDK 环境变量

```
hduser@namenode1:~$ cat /etc/profile.d/java.sh
JAVA_HOME=/home/hduser/jdk
HADOOP_HOME=/usr/local/hadoop
STORM_HOME=/home/storm
export JAVA_HOME HADOOP_HOME STORM_HOME
PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$STORM_HOME/bin/
```

配置 hadoop 环境变量

```
hduser@namenode1:~$ cat /etc/profile.d/hadoop.sh
HADOOP_HOME=/usr/local/hadoop
PATH=$HADOOP_HOME/bin:$PATH:/usr/local/hadoop/sbin/
export HADOOP_HOME PATH
```

验证 HADOOP 环境变量是否生效

```
hduser@namenode1:~$ echo $HADOOP_HOME
/usr/local/hadoop
```

查看 hadoop 版本

```
hduser@namenode1:~$ hadoop version
Hadoop 2.2.0
Subversion Unknown -r 1556437
```

```
Compiled by hadoop on 2014-01-08T04:16Z
Compiled with protoc 2.5.0
From source with checksum 79e53ce7994d1628b240f09af91e1af4
This command was run using /usr/local/hadoop-2.2.0/share/hadoop/common/hadoop-common-2.2.0.jar
```

验证 JDK 环境变量是否生效

```
hduser@namenode1:~$ echo $JAVA_HOME
/home/hduser/jdk
hduser@namenode1:~$ java -version
java version "1.7.0_03"
OpenJDK Runtime Environment (IcedTea7 2.1.7) (7u3-2.1.7-1)
OpenJDK 64-Bit Server VM (build 22.0-b10, mixed mode)
```

2.4 修改配置文件

指定 slaves 节点

```
hduser@namenode1:/usr/local/hadoop/etc/hadoop$ cat slaves
namenode2
datanode1
```

配置 core-site.xml

```
hduser@namenode1:/usr/local/hadoop/etc/hadoop$ cat core-site.xml
<configuration>
 <property>
   <name>hadoop.tmp.dir</name>            #属性用于定义 Hadoop 的临时目录至/hadoop/temp/
  <value>/hadoop/temp</value>
 </property>

   <property>
      <name>fs.default.name</name>       #定义 HDFS 的名称节点和其默认的文件系统
      <value>hdfs://namenode1:8020</value>
      <final>true</final>
   </property>
</configuration>
```

配置 mapred-site.xml

```
hduser@namenode1:/usr/local/hadoop/etc/hadoop$ catmapred-site.xml
<configuration>
  <property>
    <name>mapred.job.tracker</name>    #可以接受多个以逗号分隔路径列表作为其值，并会以轮流的方式将数据分散存储在这些文件系统上，因此指定位于不同磁盘上的多个文件系统路径可以分散数据 I/O
    <value>namenode1:8021</value>      #
    <final>true</final>
  </property>

<property>
  <name>mapred.reduce.tasks</name>    #定义每任务的 reduce 数量
  <value>2</value>
</property>
</configuration>
```

配置 hdfs-site.xml

```
hduser@namenode1:/usr/local/hadoop/etc/hadoop$ cat hdfs-site.xml
<configuration>
<property>
  <name>dfs.replication</name>          #定义 dfs 节点个数，这里我们就有一个 datanode
  <value>1</value>
</property>

<property>
  <name>dfs.data.dir</name>             #HDFS 文件数据块的本地目录
  <value>/hadoop/data</value>
  <final>ture</final>
</property>

<property>
  <name>dfs.name.dir</name>             #HDFS 元数据的本地目录
  <value>/hadoop/name</value>
```

```
    <final>ture</final>
</property>

<property>
    <name>fs.checkpoint.dir</name>        #定义的 SecondaryNameNode 用于存储检查点文件的目录
    <value>/hadoop/namesecondary</value>
    <final>ture</final>
</property>

</configuration>
```

设置缓冲大小

Hadoop 为其 I/O 操作使用了 4KB 的缓冲区容量 ,这个值是相当保守的。在当今的硬件和操作系统上，可以安全地增大此值以提高系统性能；一般说来，128KB(131072 bytes)是比较理想的设定。如果需要，可以在 core-site.xml 中通过 io.file.buffer.size 属性进行定义--在 core-site.xml 文件中追加以下内容

```
  <property>
      <name>io.file.buffer.size</name>
       <value>4096</value>
  </property>
```

3.将配置复制至其他节点

```
hduser@namenode1:/usr/local/hadoop/etc/hadoop$   scp *hduser@datanode1:/usr/local/hadoop/etc/hadoop/
hduser@namenode1:/usr/local/hadoop/etc/hadoop$   scp *
hduser@namenode2:/usr/local/hadoop/etc/hadoop/
```

4.初始化数据节点

```
hduser@namenode1:$ hadoop namenode -format
```

5.启动并验证 Hadoop

为了方便，分别在其他节点执行 start-all.sh，较大的内在需求；而在运行着众多 MapReduce 任务的环境中，JobTracker 节点会用到大量的内存和 CPU 资源，因此，此场景中通常需要将 NameNode 和 JobTracker 运行在不同的物理主机上，也就是说如果在生产环境中，需要对服务角色进行执行对应的脚本

```
hduser@namenode1:/usr/local/hadoop/etc/hadoop$ cd/usr/local/hadoop/sbin/
hduser@namenode1:/usr/local/hadoop/sbin$ ./start-all.sh
```

使用 jps 命令分别查看不同角色服务器所运行的 hadoop 服务

```
hduser@namenode1:/usr/local/hadoop/sbin$ jps
3624 Jps
2951 NameNode
3351 ResourceManager
3125 SecondaryNameNode
hduser@namenode2:/usr/local/hadoop/etc/hadoop$ jps
31488 Jps
29784 SecondaryNameNode
31394 NodeManager
hduser@datanode1:~$ jps
32416 DataNode
462 Jps
32693 ResourceManager
328 NodeManager
32535 SecondaryNameNode
```

hdfs 文件系统的状态信息

```
hduser@namenode1:/usr/local/hadoop/logs$ hdfs dfsadmin -report
Configured Capacity: 30302740480 (28.22 GB)
Present Capacity: 18599096320 (17.32 GB)
DFS Remaining: 18599047168 (17.32 GB)
DFS Used: 49152 (48 KB)
DFS Used%: 0.00%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0

-------------------------------------------------
Datanodes available: 2 (2 total, 0 dead)

Live datanodes:
Name: 172.23.215.61:50010 (datanode1)
Hostname: datanode1
```

Decommission Status : Normal

Configured Capacity: 15151370240 (14.11 GB)

DFS Used: 24576 (24 KB)

Non DFS Used: 5851353088 (5.45 GB)

DFS Remaining: 9299992576 (8.66 GB)

DFS Used%: 0.00%

DFS Remaining%: 61.38%

Last contact: Wed Mar 05 16:56:09 CST 2014

Name: 172.23.214.47:50010 (namenode2)

Hostname: namenode2

Decommission Status : Normal

Configured Capacity: 15151370240 (14.11 GB)

DFS Used: 24576 (24 KB)

Non DFS Used: 5852291072 (5.45 GB)

DFS Remaining: 9299054592 (8.66 GB)

DFS Used%: 0.00%

DFS Remaining%: 61.37%

Last contact: Wed Mar 05 16:56:08 CST 2014

访问

查看 RM

查看HDFS

查看 HDFS

NameNode 'namenode1:8020' (active)

Started:	Thu Mar 06 13:48:23 CST 2014
Version:	2.2.0, 1556437
Compiled:	2014-01-08T04:16Z by hadoop from Unknown
Cluster ID:	CID-fdc559f1-f9ea-487b-91ae-6b83c8557e90
Block Pool ID:	BP-311149917-172.23.214.50-1394074286702

Linux Tips 之 切换用户身份 su 与 sudo 的用法与实例

作者：斑马 Linux 来源：http://zebralinux.blog.51cto.com/8627088/1369301

作中为了避免一些误操作，更加安全的管理系统，通常使用的用户身份都为普通用户，而非 root。当需要执行一些管理员命令操作时，再切换成 root 用户身份去执行。
普通用户切换到 root 用户的方式有：su 和 sudo。

1，su -

（su 为 switch user，即切换用户的简写）
格式：su -l USERNAME（-l 为 login，即登陆的简写）
-l 可以将 l 省略掉，所以此命令常写为 su - USERNAME

如果不指定 USERNAME（用户名），默认即为 root，所以切换到 root 的身份的命令即为：su -root 或是直接 su -

实例 1 :普通用户 user1 知道 root 账户登录密码 ,要求用户 user1 在不注销登录的前提下查看/etc/shadow 文件。
如下图，试图查看文件/etc/shadow 时，提示拒绝访问，此时使用 su - 命令切换成 root 身份后，即可正常查看。

```
[user1@localhost ~]$ tail /etc/shadow
tail: cannot open `/etc/shadow' for reading: Permission denied
[user1@localhost ~]$ su -
Password:
[root@localhost ~]# tail /etc/shadow
mockbuild:!!:16121:0:99999:7:::
nginx:!!:16122::::::
```

之后，通过命令 exit 或 logout，或者是快捷键 Cry+D 即可返回原用户身份。

2：su - 与 su

通过 su 切换用户还可以直接使用命令 su USERNAME，与 su - USERNAME 的不同之处如下：
su - USERNAME 切换用户后，同时切换到新用户的工作环境中

su USERNAME 切换用户后，不改变原用户的工作目录，及其他环境变量目录
如下图，显示两个命令的执行结果：

```
[user1@localhost ~]$ su -
Password:
[root@localhost ~]# pwd
/root
[root@localhost ~]# env | grep root
USER=root
MAIL=/var/spool/mail/root
PATH=/usr/lib64/qt-3.3/bin:/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
PWD=/root
HOME=/root
LOGNAME=root
XAUTHORITY=/root/.xauthGthMkW
[root@localhost ~]# logout
[user1@localhost ~]$ su
Password:
[root@localhost user1]# pwd
/home/user1
[root@localhost user1]# env | grep user1
USER=user1
PATH=/usr/local/apache2/bin:/usr/lib64/qt-3.3/bin:/usr/local/apache2/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/user1/bin
MAIL=/var/spool/mail/user1
PWD=/home/user1
LOGNAME=user1
```

3，sudo

使用 su 切换用户时需知晓对应用户的登陆密码，即若切换成 root 用户身份，需知道 root 用户的登陆密码。作为 root 用户管理员，如何授权其他普通用户，在不需要知晓 root 密码的情况下，执行 root 权限的命令操作？此时即可使用 sudo。

格式：sudo -u USERNAME COMMAND

当普通用户通过 sudo 以 root 用户执行命令时，sudo 后面的 -uUSERNAME 可省略，即 sudo COMMAND 即意为 sudo 以 root 用户执行

默认情况下，系统只有 root 用户可以执行 sudo 命令。需要 root 用户通过使用 visudo 命令编辑 sudo 的配置文件/etc/sudoers，才可以授权其他普通用户执行 sudo 命令。

如下图，假如使用普通用户帐号 user4 通过 sudo 以 root 用户身份执行命令 tail /etc/shadow 时，即被提示：user4 未被定义在 sudoers 文件中，无法执行此命令。

```
[user4@localhost ~]$ sudo tail /etc/shadow
[sudo] password for user4:
user4 is not in the sudoers file.  This incident will be reported.
```

4，sudoers

sudo 的配置文件为：/etc/sudoers。

sudoers 文件中允许指定用户在不需要知道 root 用户的登陆密码的情况下，可以以 root 用户身份运行各种命令。此文件必须使用 visudo 命令编辑配置。(visudo 命令可以提供 basic sanitychecks 和 check for parse errors，即提供快速的正确性有效性检查，以及语法检查功能)

查看 sudores 文件，其中有一行如下图，定义了允许 root 用户从任何主机登陆，使用 sudo 可以切换成任何用户的身份，执行所有命令。

```
[root@localhost ~]# cat /etc/sudoers | grep root
## the root user, without needing the root password.
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
```

用户帐号　登陆来源的主机名　可切换的身份　可执行的命令

查看 sudoers 文件，其中有两行如下图，定义了组可以使用 sudo 命令的配置。

```
## Allows people in group wheel to run all commands
# %wheel        ALL=(ALL)       ALL

## Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL
```

允许组wheel的组成员运行所有命令

允许组wheel的组成员运行所有命令，并且不需要密码

实例 2：设置普通用户 user4，使其可以使用 sudo 命令以 root 用户身份修改其他所有用户登录密码，但不能修改 root 用户登陆密码

未被授权前，user4 使用 sudo 以 root 用户修改 user1 的密码时，提示 user4 未被定义在 sudoers 文件中，无法执行，并且此事件将被报告给。如下图：

```
[user4@localhost ~]$ passwd user1
passwd: Only root can specify a user name.
[user4@localhost ~]$ sudo passwd user1
[sudo] password for user4:
user4 is not in the sudoers file.  This incident will be reported.
```

执行 visudo 命令，编辑 sudoers 文件，添加一行：用户帐号为 user4；可以从任何主机登陆，执行三条命令（以 root 身份执行 passwd 命令后面不加用户名时，即代表修改 root 用户本身的密码，此即为第一条命令的作用），如下图，即可实现 user4 可以修改除 root 用户之外的其他所有用户的登录密码。

```
[root@localhost ~]# visudo
...
##
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
user4   ALL=(root)      !/usr/bin/passwd, /usr/bin/passwd [A-Za-z]*, !/usr/bin/passwd root
```

不允许执行passwd　　不允许执行passwd root

允许执行passwd [A-Za-z]*

之后，user4 通过 sudo 以 root 用户身份即可成功修改 user1 的密码（而不需要知道 root 的密码，只需要输入自己的密码即可），同时无法修改 root 的密码，如下图：

```
[user4@localhost ~]$ sudo passwd user1
Changing password for user user1.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[user4@localhost ~]$ sudo passwd
Sorry, user user4 is not allowed to execute '/usr/bin/passwd' as root on localhost.localdomain.
[user4@localhost ~]$ sudo passwd root
Sorry, user user4 is not allowed to execute '/usr/bin/passwd root' as root on localhost.localdomain.
```

实例 3：设置组 Administrators 内所有成员都可以通过 sudo 以 root 用户身份执行所有命令，且不需要验证本身的账户密码。

通过 visudo 命令编辑 sudoers 文件，添加如下一行，即完成配置。

```
[root@localhost ~]# visudo

## Sudoers allows particular users to run various commands as
## Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL
%Administrators ALL=(ALL)       NOPASSWD: ALL
```

之后，成员一 user1，即可通过 sudo 以 root 用户执行所有命令，且不需要验证本身账户密码。如下图：

```
[root@localhost ~]# cat /etc/group | grep Administrator
Administrators:x:504:user1,user2,user3
[root@localhost ~]# su - user1
[user1@localhost ~]$ sudo useradd user10
[user1@localhost ~]$ tail -n -1 /etc/passwd
user10:x:510:512::/home/user10:/bin/bash
[user1@localhost ~]$ sudo passwd root
Changing password for user root.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

附：man 文档中 su 和 sudo 的解释：

su - run a shell with substitute user andgroup IDs

以替代的用户运行 shell。（即 su 之后，在当前 shell 上的用户身份已转变）

sudo - excute a command as another user.

sudo allows a permitted user to execute acommand as the superuser or another user, as specified by security policy.

以其他用户身份执行命令。sudo 依照安全策略中指定，允许授权用户以超级用户或是其他用户身份执行命令。（即 sudo，只是临时以其他用户身份执行命令，并不会切换身份）

su -c

当然，su 也可以在不切换用户身份的情况下，临时以其他用户执行命令。

通过选项-c，即可使用 root 身份临时执行命令，如下图：

```
[user1@localhost ~]$ su - -c "mkdir /tmp/rootfolder"
Password:
[user1@localhost ~]$ ll -d /tmp/rootfolder
drwxr-xr-x. 2 root root 4096 Mar  6 22:07 /tmp/rootfolder
```

/bin/su -

同时，也可以通过配置 sudoers 文件，授权其他普通用户，可以切换成其他用户身份去执行命令，而不必每次都加上 sudo。如下图，只需在 sudoers 文件中定义普通用户 user4。

```
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
user4   ALL=(root)      NOPASSWD: /bin/su -
```

之后，用户 user4 只需执行一次 sudo su - 即可切换成 root 身份了。

```
[user4@localhost ~]$ sudo su -
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root),490(sfcb) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

注：实例环境为 Vmware Workstation 9、CentOS 6.4

Haproxy 负载均衡平滑上线，下线后端网站服务器方案

作者： jk88 来源：http://jiechao2012.blog.51cto.com/3251753/1359511

#智能判断，网站 web02 服务器是否在线上，如果在 haproxy 线上，就无需要在挂到 haproxy 上。

#现在 3 台 web 网站服务器还在线上

		Queue			Session rate			Sessions					Bytes	
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out
☐	web01	0	0	-	1	8		0	6	-	23 413	23 413	35 437 918	479 941 949
☐	web02	0	0	-	2	12		1	7	-	39 023	39 023	56 277 991	811 597 728
☐	web03	0	0	-	3	13		1	6	-	39 023	39 023	59 847 123	795 891 355
	Backend	0	0		6	33		2	14	0	101 459	101 459	151 563 032	2 087 431 032

#现在下线 web02 网站服务器.

#在 haproxy 监控页面，看到 web02 网站服务器已经不在线上，不对用户提供请求了。

		Queue			Session rate			Sessions					Bytes	
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out
☐	web01	0	0	-	1	5		0	3	-	27	27	29 038	615 047
☐	web03	0	0	-	2	9		1	3	-	45	45	40 636	927 997
	Backend	0	0		4	14		1	4	0	72	72	69 674	1 543 044

#现在把 web02 网站服务器挂到 haproxy 负载均衡上.

```
本机Haproxy负载均衡路径为: /usr/local/haproxy/

****************** Haproxy tool ******************
*                                                *
* (1)  平滑启动Haproxy                           *
* (2)  关闭Haproxy                               *
* (4)  查看Haproxy运行进程数                     *
* (5)  查看TCP连接状态                           *
* (10) Haproxy平滑下线方案                       *
* (11) Haproxy平滑上线线方案                     *
* (0)  退出本程序                                *
*                                                *
**************************************************
请输入对应数字: 11
```

```
      本机Haproxy-主站平滑上线方案

**************Haproxy 上线主机列表****************
*                                                *
* (1)  平滑上线web01:[redacted]主机              *
* (2)  平滑上线web02:[redacted]主机              *
* (3)  平滑上线web03:[redacted]主机              *
* (0)  退出本程序                                *
*                                                *
**************************************************
请输入对应数字: 2
开始平滑上线web02网站服务器.....................................
[WARNING] 046/113427 (2889) : parsing [/usr/local/haproxy/etc/haproxy.cfg:67] :
[WARNING] 046/113427 (2889) : Proxy 'admin_stats': in multi-process mode, stats
[WARNING] 046/113427 (2889) : Proxy 'admin_stats': stats admin will not work co
上线完成,回车返回!
```

#web02 上线完成,到 haproxy 监控页面看下，是否在 haproxy 负载均衡上面。

		Queue			Session rate			Sessions					Bytes		Denied	
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp
☐	web01	0	0	-	1	1		1	1	-	1	1	0	0		0
☐	web02	0	0	-	1	1		0	1	-	2	2	552	68 152		0
☐	web03	0	0	-	1	2		1	1	-	2	2	271	36 945		0
	Backend	0	0		3	4		2	2	0	5	5	823	105 097	0	0

Choose the action to perform on the checked servers : Apply

快速实验 win2003NLB 负载均衡

作者： 阿蒙爱吹牛 来源：http://ansed.blog.51cto.com/6510194/1377663

web 服务器是 2003 系统，为了保证业务的正常运行，一直有另一台相同的 web 服务器做冷备。然而冷备的缺点就是假如生产机宕掉或者服务停掉，就需要手工的将冷备的机器切换上线；这样便造成了业务正常运行的断点。

为保证业务的正常连续运行，需要做负载均衡同时也是双机热备，此次先在非生产环境做好实验。

一、环境描述：

VMware Workstation 版本：10.0.0 build-1295980

操作系统：win2003 企业版

每台虚机需要两块网卡，一块网卡用于绑定群集 ip，实验中称为公网网卡；另一块用于节点间相互通讯，

试验中称为内网网卡。

计算机 a：公网地址：192.168.137.19

内网地址：100.0.0.1

计算机 b：公网地址：192.168.137.29

内网地址：100.0.0.2

集群 ip 地址：192.168.137.39

两台虚机的管理员 Administrator 密码需一致，不能为空，否则会在添加主机到群集时需添加用户名/密码，出现未知错误。

二、具体实施

1、先做好一台 win2003 的虚拟机，需要注意的是网卡设置成桥接模式，存放虚机的文件夹命名为 cp1。

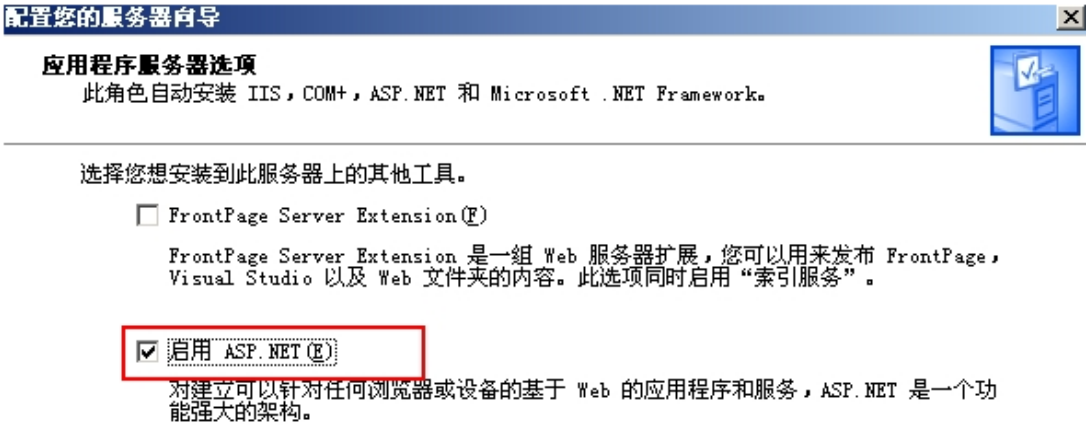
2、复制做好的虚机

3、打开虚机 1，需再添加一块网卡，同样是桥接模式。

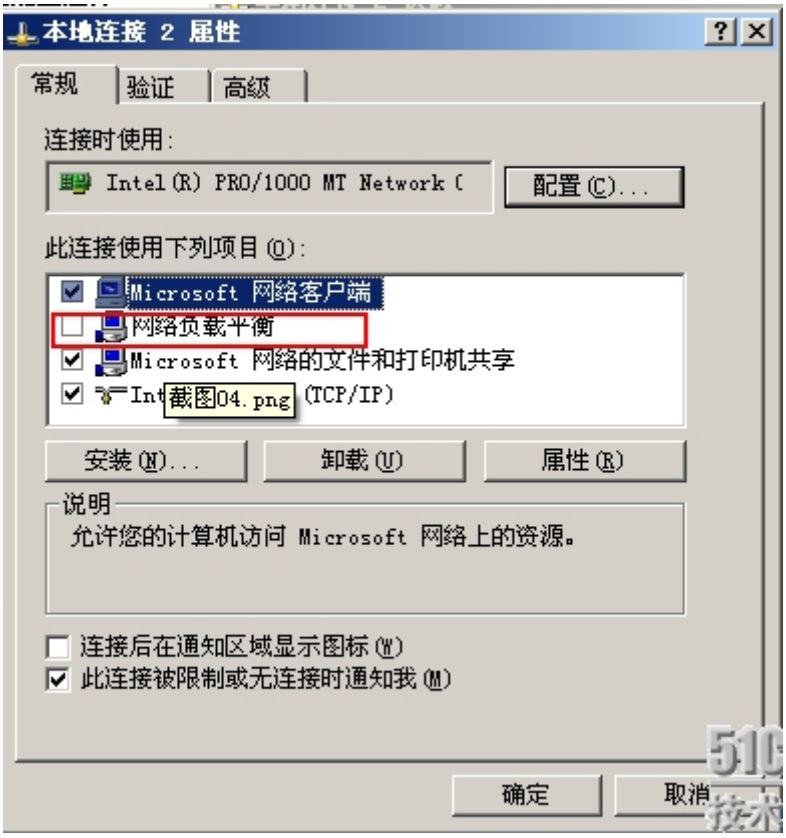
4、打开虚机 2（即复制过的虚机），需先添加两块网卡，同样是桥接模式。然后再把原先的网卡删掉。

5、开机，提示如下选项时，选择复制

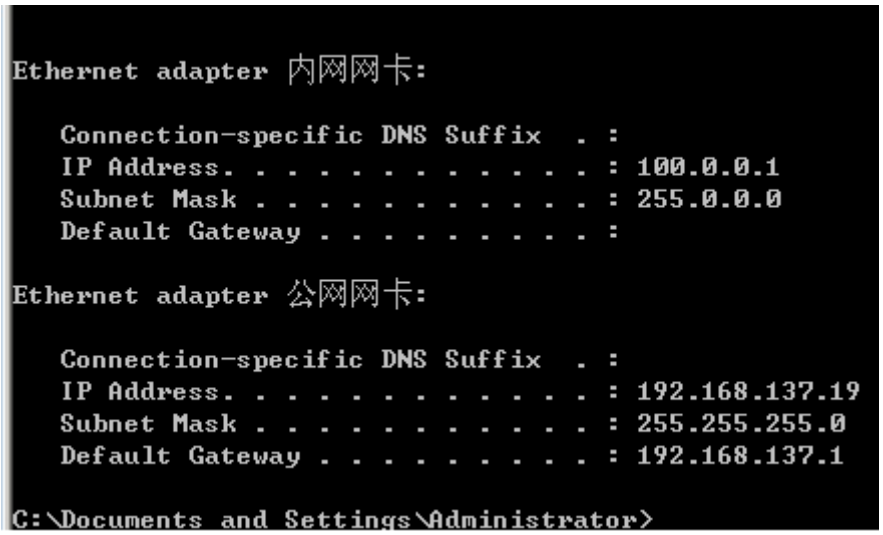
6、配置管理员密码并安装 IIS，记得选择 asp.net，否则容易出错误。



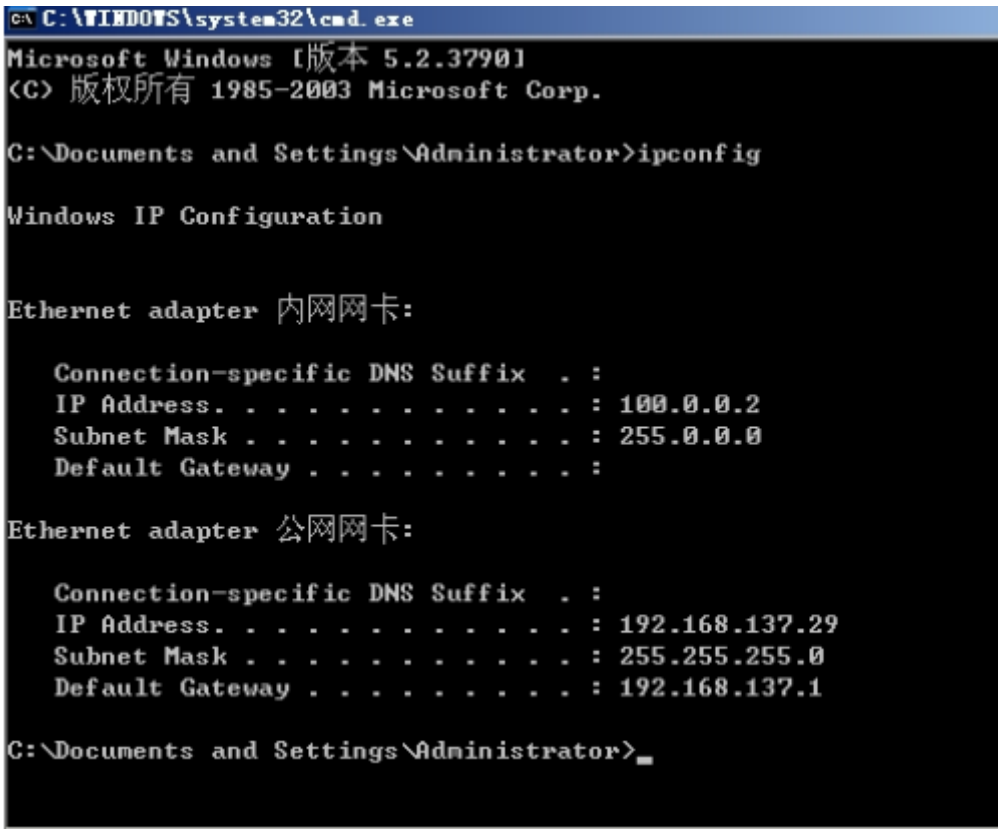
7、配置网卡地址，不要选择网络负载平衡



计算机 a 的地址



计算机 b 的地址



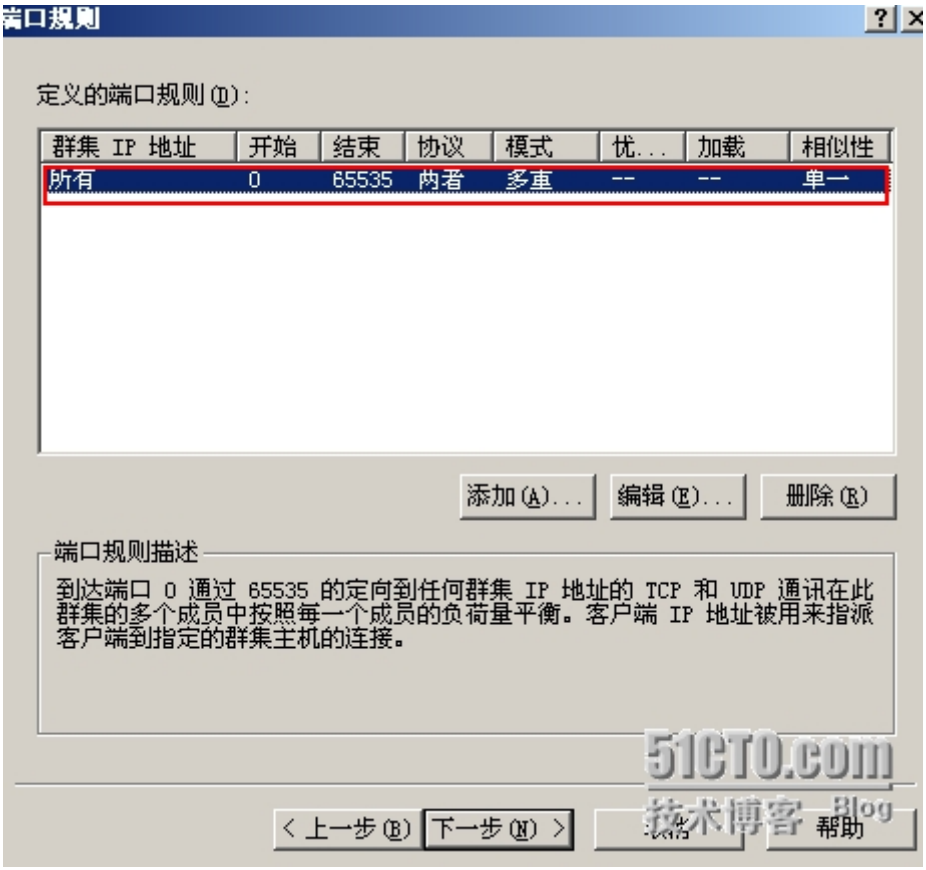
8、打开网络负载平衡管理器

9、选择新建

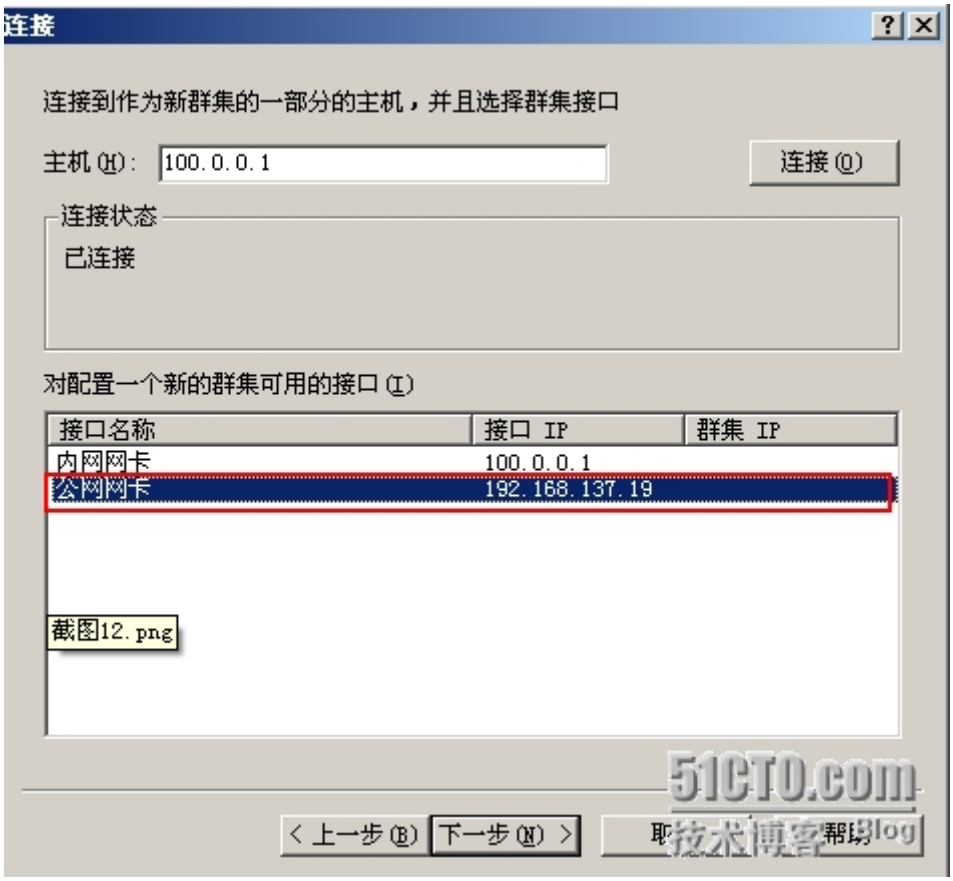
10、注意群集参数选择多播

11、默认下一步

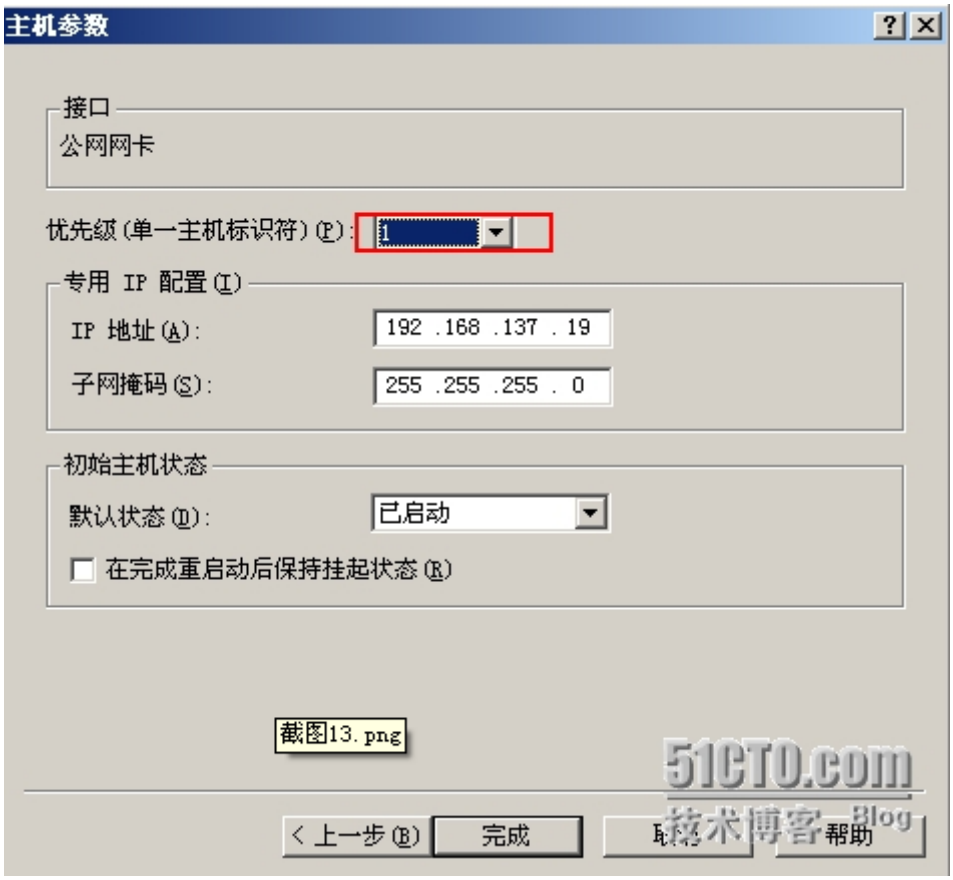
12、将默认的这条规则去掉



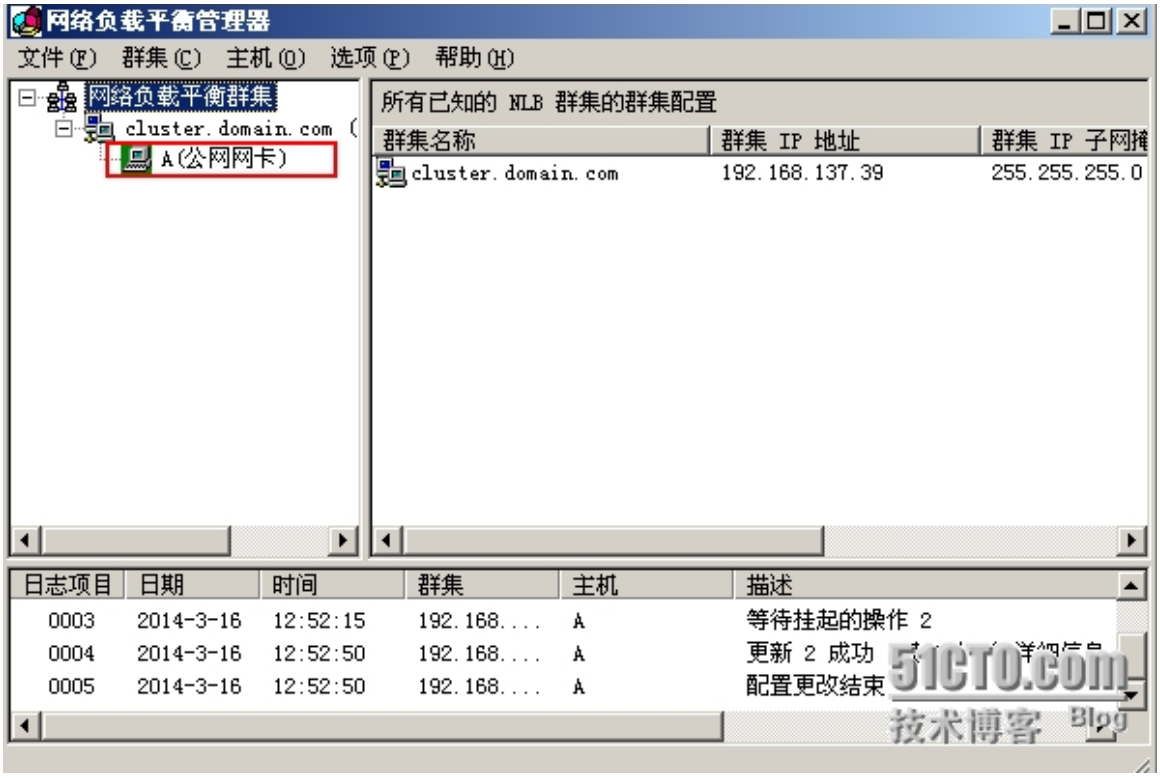
13、输入本机的内网网卡地址，选择公网网卡作为群集的网卡。



14、优先表示符的意思是机器的优先替代顺序，1 号机宕掉的话 2 号机会替代上来，以此类推。



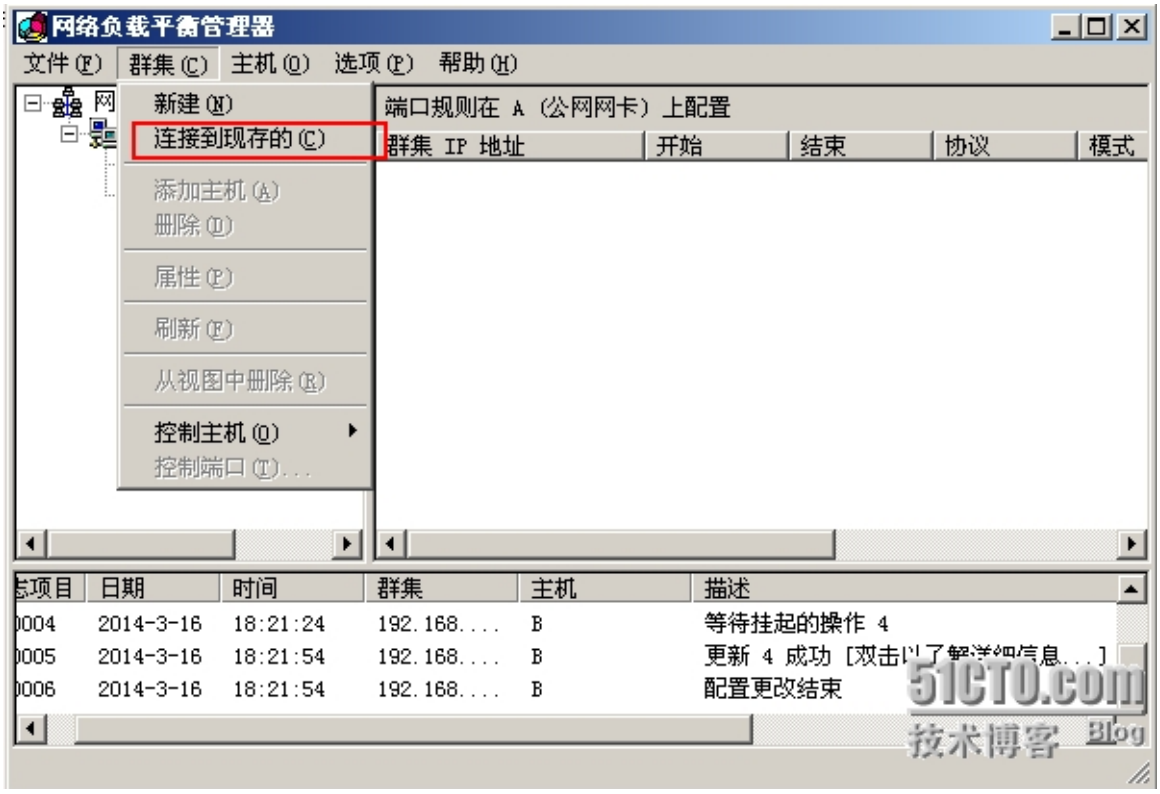
15、点击“完成”后，等待配置结果。



显示绿色配置成功。

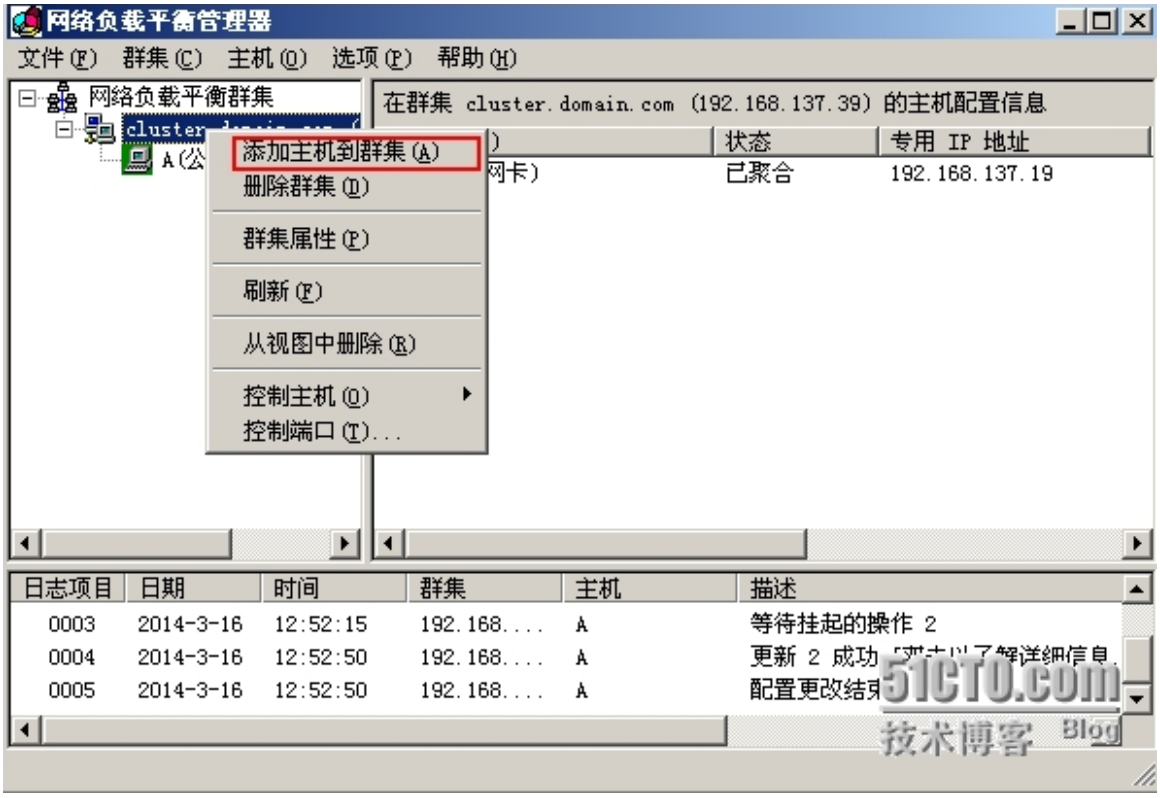
16、添加另外一台主机，进入另外一台计算机 b 上，打开网络负载平衡管理器，选择连接到现存的集群。

（一定是进入另一台计算机 b 添加，如果还在 a 上添加会报通讯借口错误）

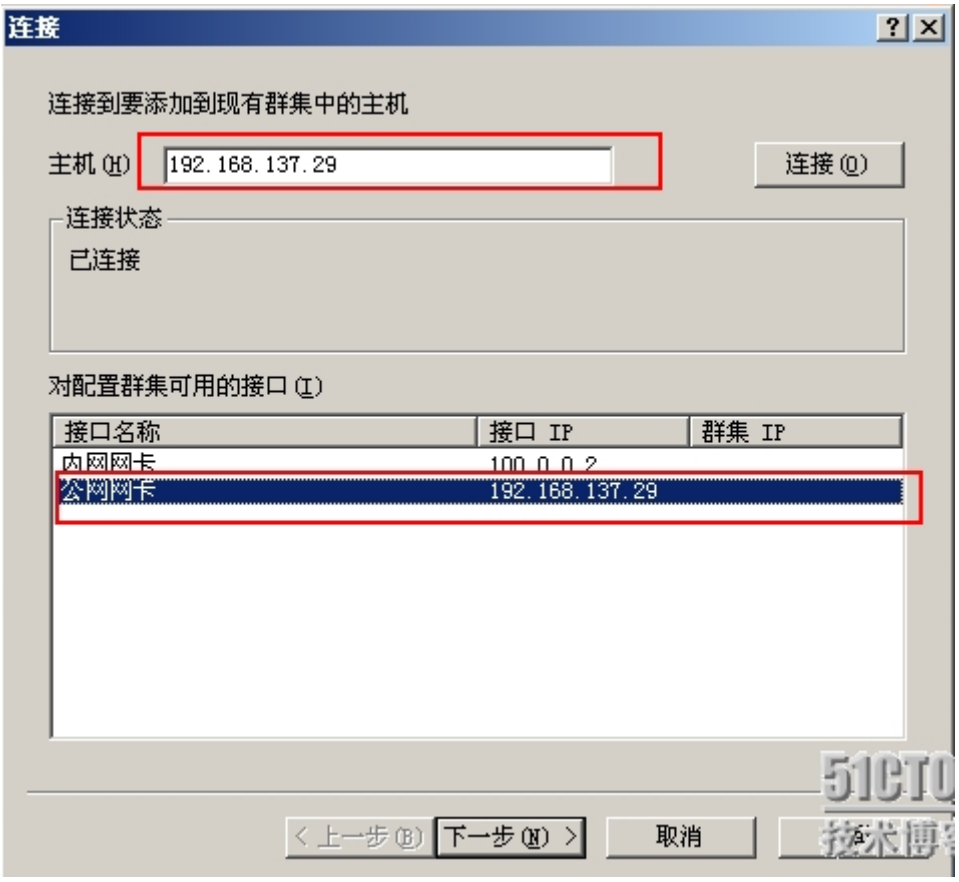


17.输入计算机 a 的地址，添加集群

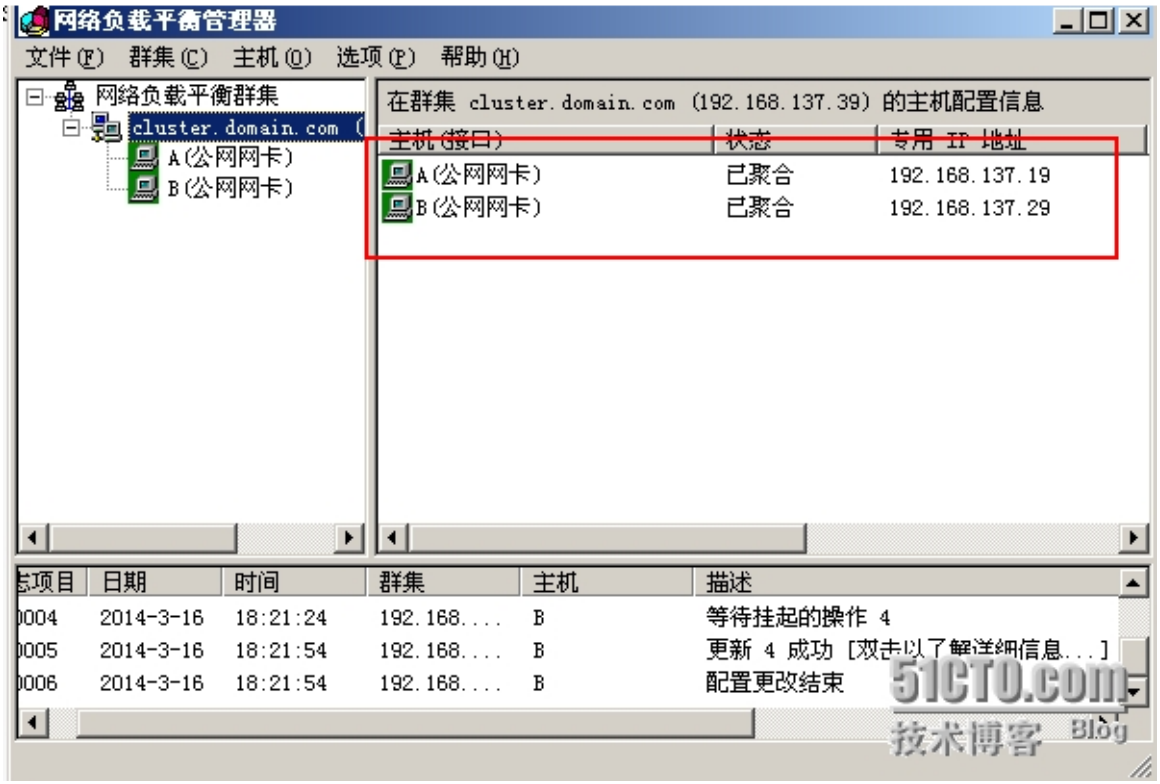
18、然后选择添加主机到集群



19、选择计算机 b 的公网网卡地址并连接，在可用接口中选择公网地址：



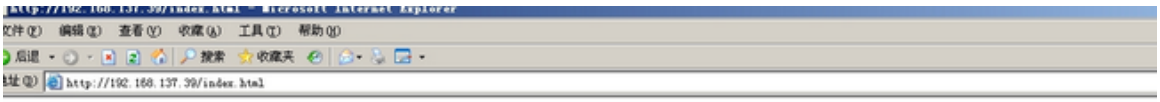
20、等待时间比较长，结束



21、测试

分别在两台计算机建立网站，网站文件夹新建文本文档，重名为“index.html”，a 的 index.html 内容

为 a 的公网地址：192.168.137.19；b 的 index.html 内容为 b 的公网地址：192.168.137.29。浏览器输入 192.168.137.39，显示内容为 a 的公网地址：192.168.137.19



22、禁用 a 的公网网卡，再次刷新页面，显示内容为 b 的公网地址：192.168.137.29

三、总结

,以上只是测试的时候，为了验证网络负载平衡的效果，两个网站的内容不一致，而在正式应用的时候，网络负载平衡群集的每个节点计算机的内容将是一致的，这样，不管使用那一个节点响应，都保证访问的内容是一致的。

第十步中的几个选项解释

单播

在单播模式下，NLB 服务会重新对每个节点中启用 NLB 的网卡分配 MAC 地址（此 MAC 地址称为群集 MAC 地址），并且所有的 NLB 节点均使用相同的 MAC 地址（均使用群集 MAC 地址），同时 NLB 会修改所有发送的数据包中的源 MAC 地址，这样就导致交换机不能将此群集 MAC 地址绑定在某个端口上。

工作在单播模式下的 NLB 可以在所有网络环境下正常运行（兼容性最好），但是由于它的工作特性，具有以下两个限制：

1.由于 NLB 所使用的群集 MAC 地址没有绑定在某个具体的交换机端口上，所以所有的 NLB 通讯均通过在交换机的所有端口上广播进行，而不管此端口是否连接了 NLB 节点，这造成了额外的网络流量负担；

2.由于所有的 NLB 节点具有相同的 MAC 地址，NLB 节点之间不能通过自己原有的专用 IP 地址进行通讯。（例如我们见得最多的就是节点之间就无法 ping 通）

多播

在多播模式下，NLB 不会修改 NLB 节点启用 NLB 的网络适配器的 MAC 地址，而是为它再分配一个二层多播 MAC 地址专用于 NLB 的通讯（此 MAC 地址称为群集 MAC 地址），这样 NLB 节点之间可以通过自己原有的专用 IP 地址进行通讯。但是在多播模式中，NLB 节点发送的针对群集 IP 地址/MAC 地址 ARP 请求的 ARP 回复会将群集 IP 地址映射到多播 MAC 地址，而许多路由器或者交换机（例如，港湾和思科的某些产品）会拒绝这一行为。如何解决呢？方法是手工在路由器或交换机上添加静态映射，将群集 IP 地址映射到群集的多播 MAC 地址。

此外，Windows Server 2003 提供了一个新的特性，称为 IGMP 多播，它可以通过使用 IGMP 协议支持来使交换机只将 NLB 通讯发送到连接 NLB 节点的端口，而不是所有交换机端口。但是此特性必须要求交换机支持 IGMP 侦听，并且要求群集工作在多播模式下。

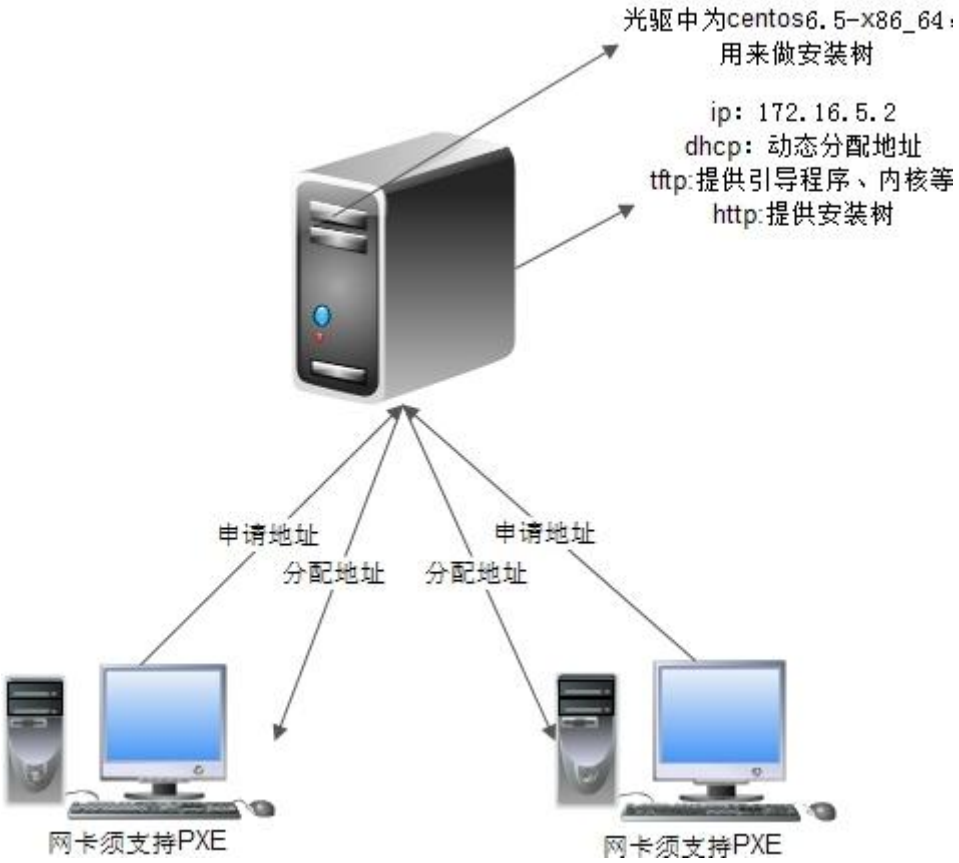
Linux—图解 PXE 实现全自动安装系统

作者：海闊天空　　来源　：http://il23f.blog.51cto.com/8620950/1376421

1、安装背景：

在实际工作中，我们经常会遇到这样的情况：想要安装 Linux 但是计算机没有光驱，或者是有大批量的计算机需要同时安装 Linux，如果通过光驱的方式一个个安装，不仅效率低，也不利于维护。这是时候你就需要 PXE 的强大功能了。本文就简单的图解一下 PXE 的安装流程。

2、本博文中 PXE 自动安装硬件架构如下图，DHCP、TFTP、HTTP 都在 172.16.5.2 这台服务器上。



3、自动安装原理：

1.客户机从自己的 PXE 网卡启动，向本网络中的 DHCP 服务器索取 IP

2.DHCP 服务器返回分给客户机 IP

3.客户机向本网络中的 TFTP 服务器索取文件

4.客户机取得 bootstrap 文件后之执行引导文件完成引导

5.然后读取配置文件，通过 TFTP 服务器加载内核和文件系统

6.进入安装画面，此时可以通过选择 HTTP、FTP、NFS（这里以 http 为例）方式进行安装

从这里我们不难得到实现 PXE 网络安装必需的 4 个要素

1.客户机的网卡必须为 PXE 网卡

2.网络中必须要有 DHCP 和 TFTP 服务器，当然这两个服务器可以是同一台物理主机

3.所安装的操作系统必须支持网络安装。

4.必须要有 FTP,HTTP,NFS 至少一个服务器，当然也可以和 DHCP 和 TFTP 服务器同为一台物理主机

4、PXE 配置流程图：

5、具体实现如下：

1、安装 dhcp、自定义作用域

```
yum install dhcp -y
[root@localhost ~]# rpm -ql dhcp
/etc/dhcp
/etc/dhcp/dhcpd.conf
subnet 172.16.0.0 netmask 255.255.0.0 {
```

```
        range 172.16.5.10 172.16.5.20;
        option routers 172.16.0.1;
        next-server 172.16.5.2;
        filename "pxelinux.0";
}
host webserver1 {
        hardware ethernet 00:0C:29:8C:C8:A4;
        fixed-address 172.16.5.100;
        option routers 172.16.0.1;
        option domain-name "http://il23f.blog.51cto.com";
        option domain-name-servers 172.16.0.1,8.8.8.8;
}
```

验证 dhcpd 进程是否处于监听状态。

```
[root@localhost ~]# ps aux | grep dhcp
dhcpd     1708  0.0  0.8  48908  4308 ?        Ss   20:27   0:00 /usr/sbin/dhcpd -user dhcpd -group
dhcpd
root      2844  0.0  0.1 103252   828 pts/0    S+   21:51   0:00 grep dhcp
[root@localhost ~]# ss -unl | grep :67
UNCONN     0      0                  *:67                       *:*
```

2、配置 TFTP

```
yum -y install xinetd tftp-server tftp
chkconfig xinetd on
chkconfig tftp on
service xinetd start
[root@localhost ~]# ss -unl | grep :69
UNCONN     0      0                  *:69                       *:*
```

3、准备安装树

```
mkdir /var/www/html/centos6
mount --bind /media/cdrom /var/www/html/centos6
service httpd start
```

准备/var/lib/tftpboot 下文件

```
yum -y install syslinux
cp /media/cdrom/images/pxeboot/{vmlinuz,initrd.img}  /var/lib/tftpboot/
cp /media/cdrom/isolinux/{boot.msg,vesamenu.c32,splash.jpg}  /var/lib/tftpboot/
cp /usr/share/syslinux/pxelinux.0  /var/lib/tftpboot/
mkdir /var/lib/tftpboot/pxelinux.cfg
cp /media/cdrom/isolinux/isolinux.cfg /var/lib/tftpboot/pxelinux.cfg/default5、制作 ks.cfg 文件
```

5、制作 kickstart 文件 ks.cfg，放到/var/www/html 目录下

```
#version=DEVEL
# Firewall configuration
firewall --disabled
# Install OS instead of upgrade
install
# Use network installation
url --url="http://172.16.5.2/centos6"
repo --name="CentOS" --baseurl=http://172.16.5.2/centos6 -- cost=100
# Root password
rootpw --iscrypted $1$seblmxLh$CowavZZ/Le1Yc9pWSXSCV/
# System authorization information
auth  --useshadow  --passalgo=sha512
# System keyboard
keyboard us
# System language
lang en_US
# SELinux configuration
selinux --disabled
# Installation logging level
logging --level=info
# Reboot after installation
reboot
# System timezone
timezone  Asia/Shanghai
# Network information
network  --bootproto=dhcp --device=eth0 --onboot=on
# System bootloader configuration
bootloader --append="crashkernel=auto crashkernel=auto rhgb rhgb quiet quiet" --location=mbr
--driveorder="sda"
# Clear the Master Boot Record
zerombr
# Partition clearing information
clearpart --all
# Disk partitioning information
part /boot --fstype="ext4" --size=200
part / --fstype="ext4" --size=5000
part swap --fstype="swap" --size=1000
%post
echo -e 'My 51cto blot: http://il23f.blog.51cto.com\n' >> /etc/issue
sed -i '1,$s@id:[0-9]:initdefault:@id:3:initdefault:@g' /etc/inittab
[ ! -d /root/.ssh ] && mkdir /root/.ssh && chmod og=--- /root/.ssh
cat >> /root/.ssh/authorized_keys << EOF
EOF
# Enable funcd
sed -i 's@certmaster =.*@certmaster = 172.16.0.1@g' /etc/certmaster/minion.conf
```

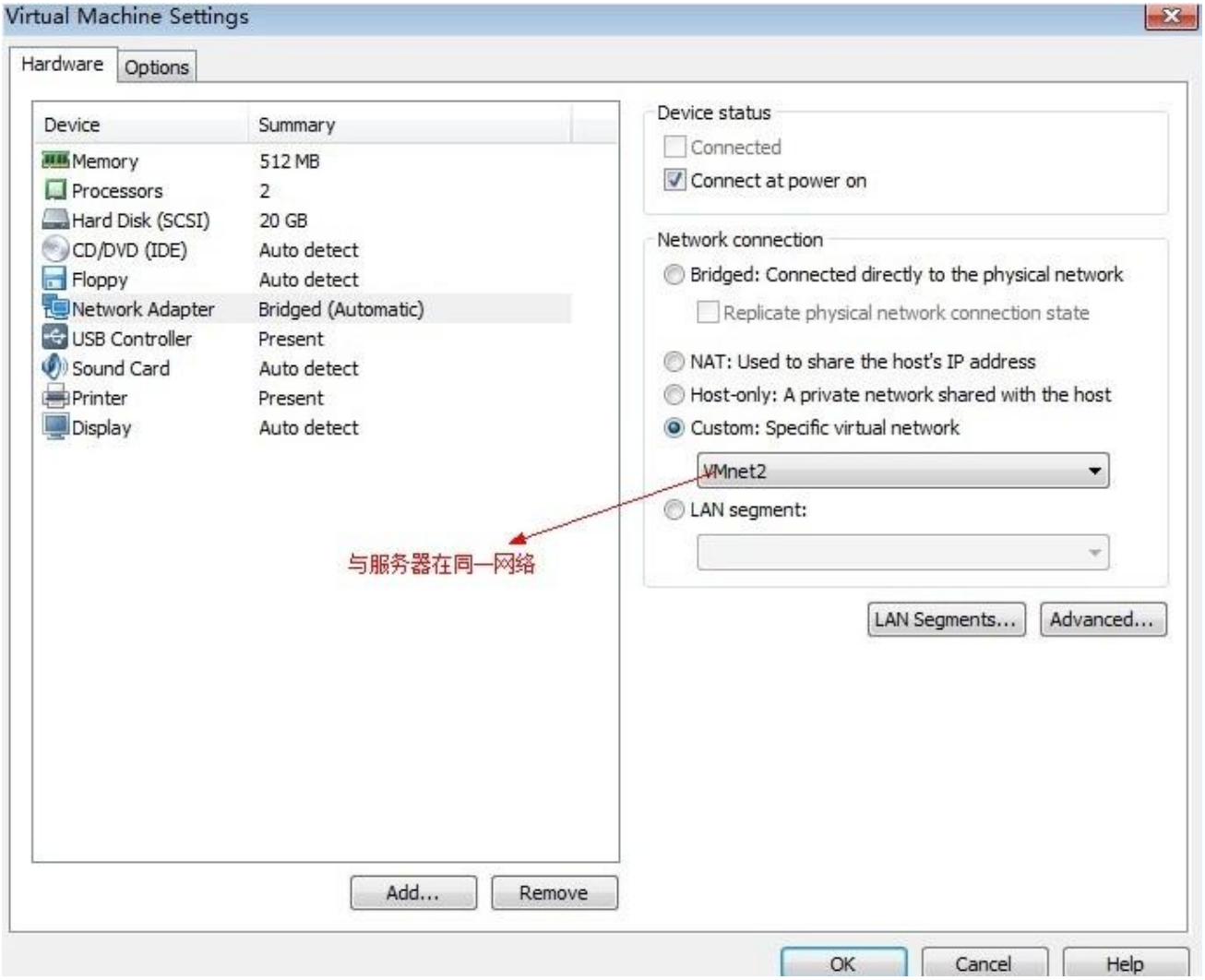
```
/sbin/chkconfig funcd off
# Set the hostname
ClientName=`ifconfig eth0 | awk '/inet addr:/{print $2}' | awk -F. '{print $NF}'`
sed -i "s@HOSTNAME=.*@HOSTNAME=client$ClientName.il23f.blog.51cto.com@g" /etc/sysconfig/networks
# set puppet agent
sed -i '/\[main\]/a server=il23f.blog.51cto.com' /etc/puppet/puppet.conf
/sbin/chkconfig puppet off
# set hosts
echo '172.16.0.1    il23f.blog.51cto.com' >> /etc/hosts
# yum repo
%end
%packages
@base
@basic-desktop
chinese-support
@client-mgmt-tools
@core
@desktop-platform
@fonts
@general-desktop
@graphical-admin-tools
@legacy-x
@network-file-system-client
@perl-runtime
@remote-desktop-clients
@x11
%end
```

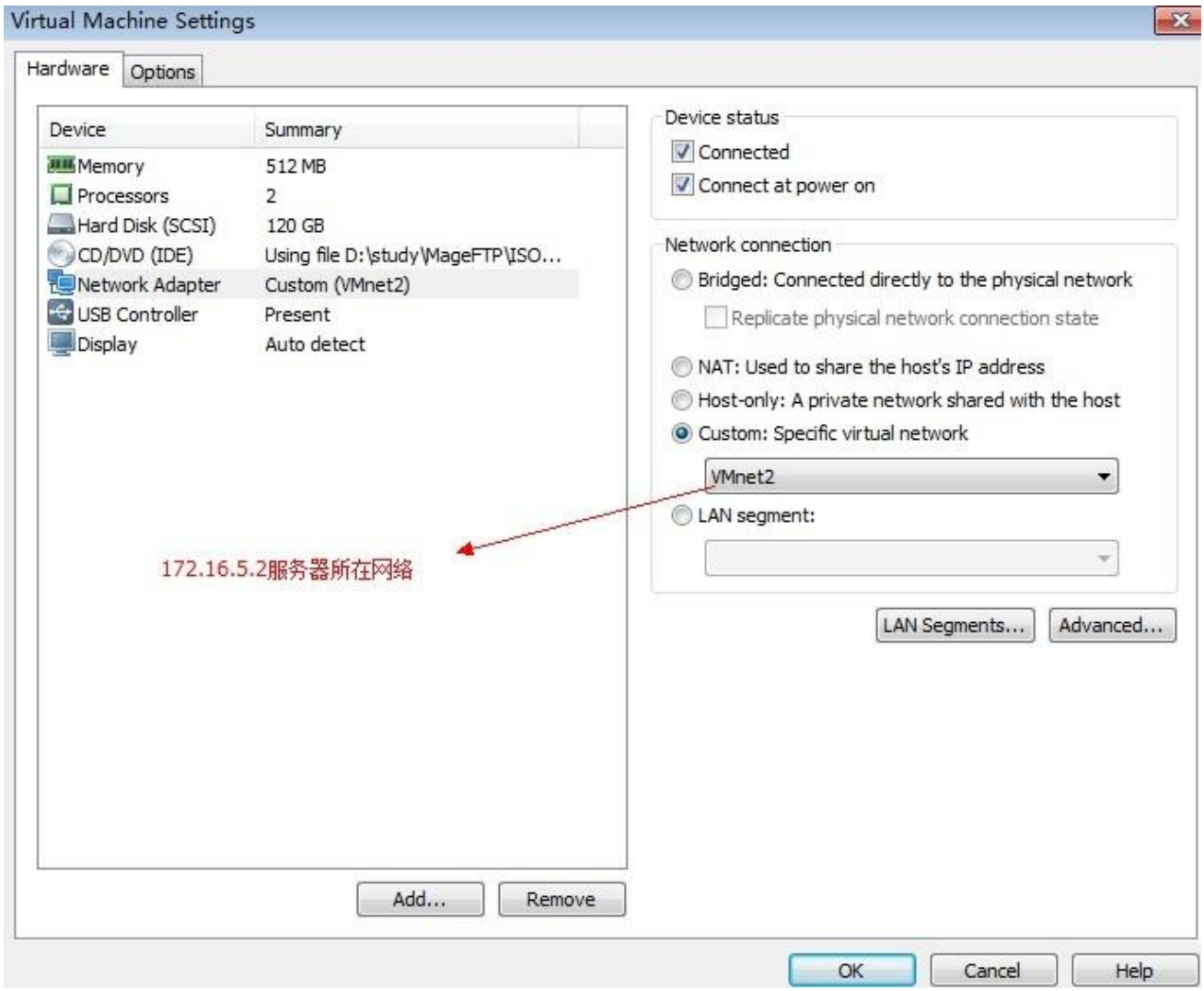
6、配置引导保证能加载 kickstart 文件，在以下文件的第 20 行的后边加上 ks=http://172.16.5.2/ks.cfg

```
[root@localhost pxelinux.cfg]# vim default
default vesamenu.c32
#prompt 1
timeout 600
display boot.msg
menu background splash.jpg
menu title Welcome to CentOS 6.5!
menu color border 0 #ffffffff #00000000
menu color sel 7 #ffffffff #ff000000
menu color title 0 #ffffffff #00000000
menu color tabmsg 0 #ffffffff #00000000
menu color unsel 0 #ffffffff #00000000
menu color hotsel 0 #ff000000 #ffffffff
menu color hotkey 7 #ffffffff #ff000000
menu color scrollbar 0 #ffffffff #00000000
label linux
```

```
menu label ^Install or upgrade an existing system
menu default
kernel vmlinuz
append initrd=initrd.img ks=http://172.16.5.2/ks.cfg
label vesa
menu label Install system with ^basic video driver
kernel vmlinuz
append initrd=initrd.img xdriver=vesa nomodeset
label rescue
menu label ^Rescue installed system
kernel vmlinuz
append initrd=initrd.img rescue
label local
menu label Boot from ^local drive
localboot 0xffff
label memtest86
menu label ^Memory test
"default" [readonly] 38L, 965C
```

6、新建一个虚拟机，网卡改到和 172.16.5.2 的服务器同一网络。





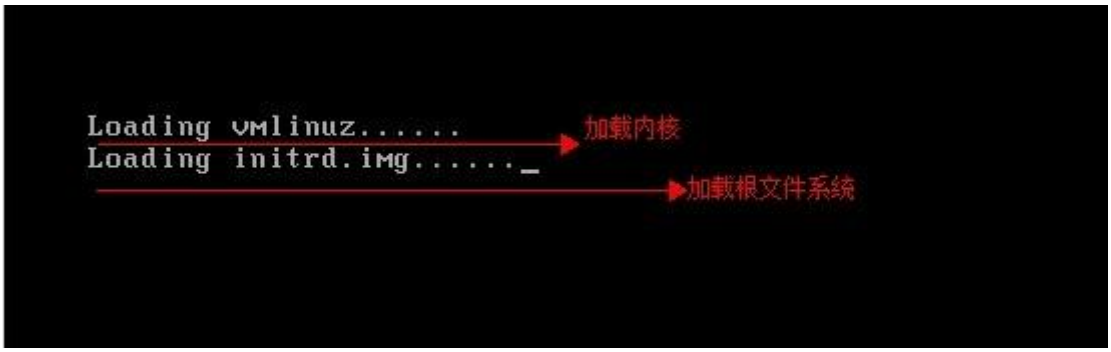
开机实现自动安装：

192.16.5.2 的服务器上查看日，已经获取到地址：

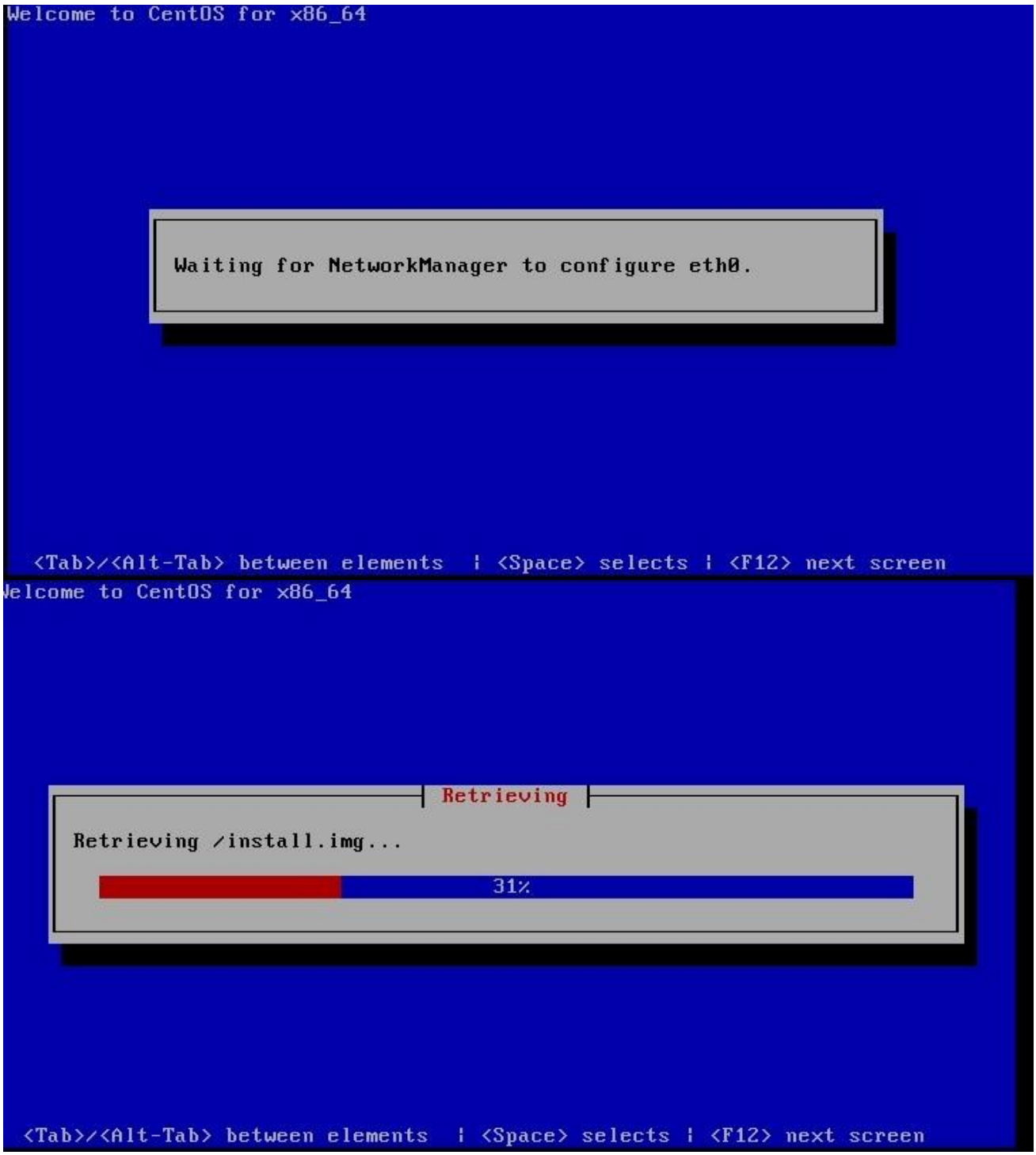
```
[root@localhost ~]# tail -f /var/log/messages
Mar 14 08:36:22 localhost dhcpd: Sending on   Socket/fallback/fallback-net
Mar 14 08:36:28 localhost xinetd[1746]: Exiting...
Mar 14 08:36:29 localhost xinetd[3144]: xinetd Version 2.3.14 started with libwr
ap loadavg labeled-networking options compiled in.
Mar 14 08:36:29 localhost xinetd[3144]: Started working: 1 available service
Mar 14 08:38:17 localhost dhcpd: DHCPDISCOVER from 00:0c:29:82:7f:96 via eth0
Mar 14 08:38:18 localhost dhcpd: DHCPOFFER on 172.16.5.14 to 00:0c:29:82:7f:96 v
ia eth0
Mar 14 08:38:19 localhost dhcpd: DHCPREQUEST for 172.16.5.14 (172.16.5.2) from 0
0:0c:29:82:7f:96 via eth0
Mar 14 08:38:19 localhost dhcpd: DHCPACK on 172.16.5.14 to 00:0c:29:82:7f:96 via
 eth0
```

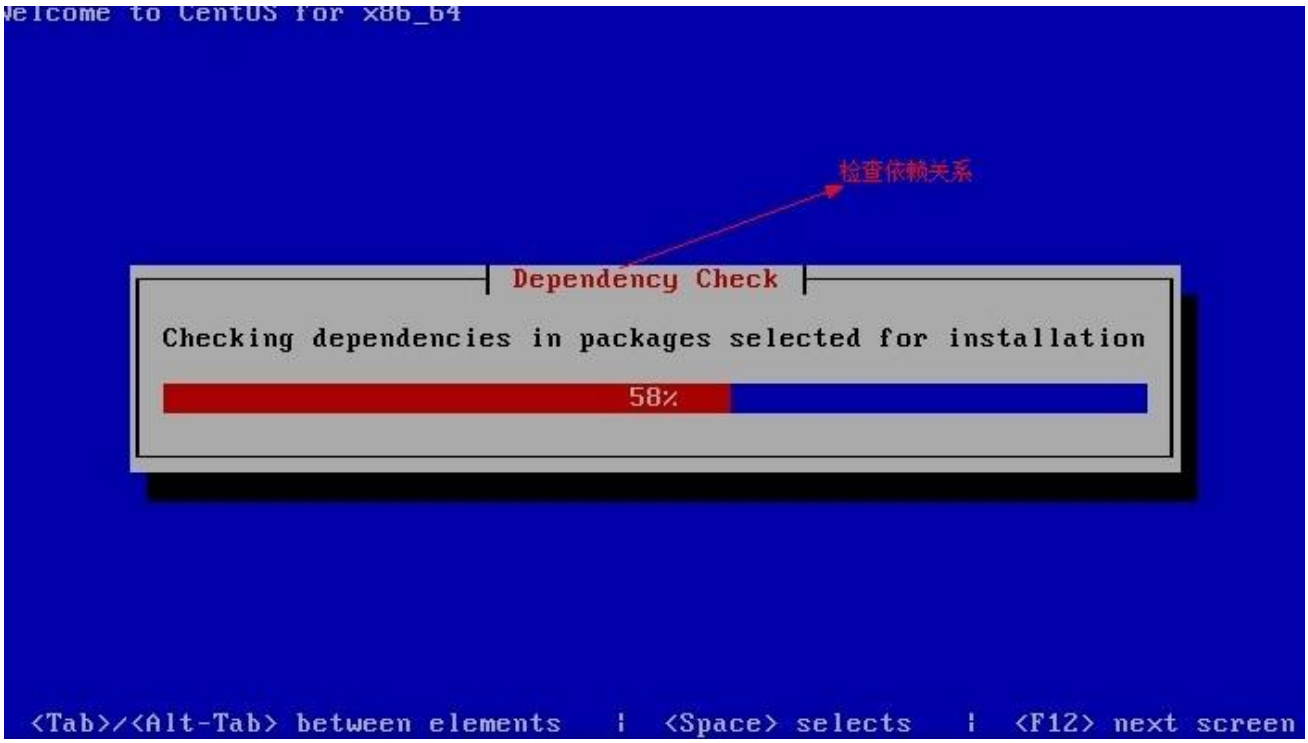
自动进入安装界面，1 分钟之后自动安装：

加载内核、根文件系统：

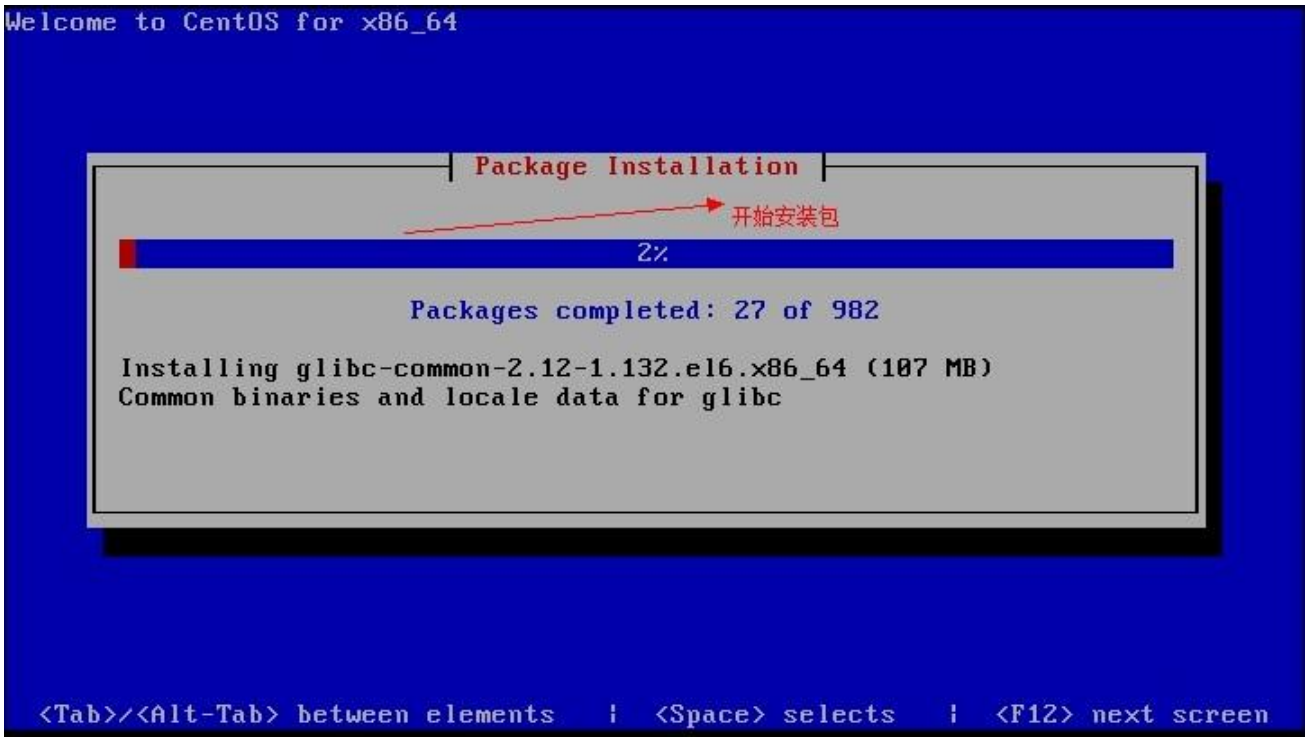


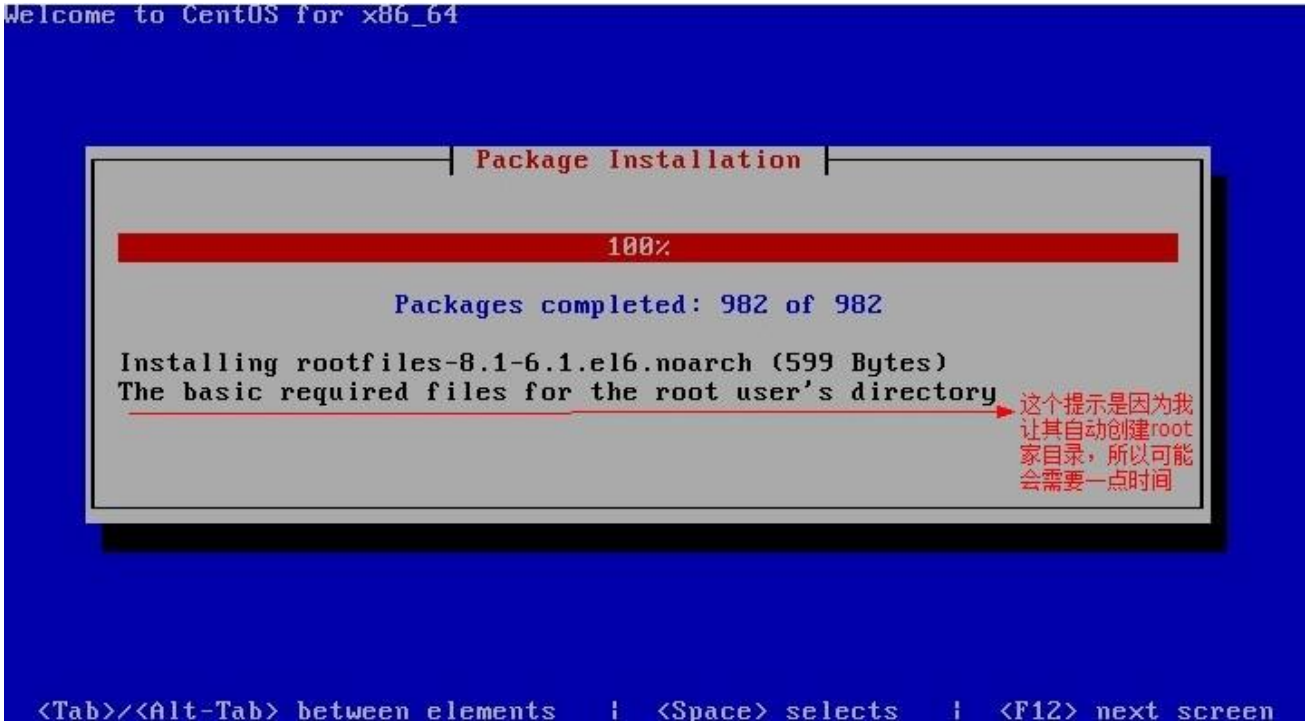
获取地址，读取配置文件，检查依赖关系等：


Welcome to CentOS for x86_64
Waiting for NetworkManager to configure eth0.
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen
Welcome to CentOS for x86_64
Retrieving
Retrieving /install.img...
31%
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen


Welcome to CentOS for x86_64
检查依赖关系
Dependency Check
Checking dependencies in packages selected for installation
58%
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen

开始安装包：


Welcome to CentOS for x86_64
Package Installation
开始安装包
2%
Packages completed: 27 of 982
Installing glibc-common-2.12-1.132.el6.x86_64 (107 MB)
Common binaries and locale data for glibc
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen



安装完成自动重启进入 login 界面：

7、安装完成进行测试：

```
[root@localhost ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:82:7F:96
          inet addr:172.16.5.14  Bcast:172.16.255.255  Mask:255.255.0.0
          inet6 addr: fe80::20c:29ff:fe82:7f96/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:684 (684.0 b)  TX bytes:1788 (1.7 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:336 (336.0 b)  TX bytes:336 (336.0 b)

[root@localhost ~]# _
[root@localhost ~]# ping 172.16.5.2 -c 5
PING 172.16.5.2 (172.16.5.2) 56(84) bytes of data.
64 bytes from 172.16.5.2: icmp_seq=1 ttl=64 time=0.599 ms
64 bytes from 172.16.5.2: icmp_seq=2 ttl=64 time=0.547 ms
64 bytes from 172.16.5.2: icmp_seq=3 ttl=64 time=0.575 ms
64 bytes from 172.16.5.2: icmp_seq=4 ttl=64 time=0.567 ms
64 bytes from 172.16.5.2: icmp_seq=5 ttl=64 time=0.484 ms

--- 172.16.5.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 0.484/0.554/0.599/0.044 ms
[root@localhost ~]# _
```

Zabbix 系列之 Zabbix 安装搭建及汉化

作者：燕十三_ 来源：http://yanshisan.blog.51cto.com/7879234/1373890

最近在研究 zabbix，在整理完成之后就有了写一下总结博客的想法，在我研究 zabbix 的时候给我很大帮助的是 it 你好，博客地址 http://itnihao.blog.51cto.com/他做的 zabbix 使用手册非常棒，很完整，在此感谢 it 你好在 zabbix 方面给大家做出的贡献，好了废话不多说，下面就开始咱们的使用 zabbix 第一步安装搭建及汉化(其实这部分完全可以去看官方文档，里面写的已经很详细很直观了，地址给大家 https://www.zabbix.org/wiki/InstallOnCentOS_6.x_RHEL_6.x)

搭建环境:Centos6.5_x86_64,Zabbix2.2.2(目前为止最新版本)，epel 源

Server 端：

1、安装开发软件包

```
yum  -y groupinstall  "Development Tools"
```

2、安装所需的依赖包

```
yum -y install httpd mysql mysql-server php php-mysql php-common php-mbstring php-gd php-odbc php-pear curl curl-devel net-snmp net-snmp-devel perl-DBI php-xml ntpdate  php-bcmath
```

3、同步服务端的时间，保持所有服务器时间一致避免出现时间不同导致的不可用的监控数据

```
ntpdate pool.ntp.rog
```

4、创建 zabbix 服务运行所需要的用户和组

```
groupadd  -g 201  zabbix
useradd  -g zabbix  -u 201 -m zabbix
```

5、初始化 mysql 服务器

```
/etc/init.d/mysqld start
```

6、创建 zabbix 运行所需要的数据库及用户权限

```
mysqladmin  -uroot -h127.0.0.1  password "123456"
mysql -uroot -h127.0.0.1 -p
create database zabbix character set utf8;
grant all privileges on zabbix.* to zabbixuser@'%' identified by 'zabbixpass';
flush privileges;
```

7、下载解压 zabbix

```
wget
http://sourceforge.net/projects/zabbix/files/ZABBIX%20Latest%20Stable/2.2.2/zabbix-2.2.2.tar.gz/download
```

```
cd /usr/src/
tar xf zabbix-2.2.2.tar.gz
```

8、将 zabbix 的初始数据导入到数据库中

```
cd zabbix-2.2.2
mysql -uzabbixuser -h192.168.239.130 -p zabbix <database/mysql/schema.sql
mysql -uzabbixuser -h192.168.239.130 -p zabbix <database/mysql/images.sql
mysql -uzabbixuser -h192.168.239.130 -p zabbix <database/mysql/data.sql
#登录数据库查看下表是否都创建成功
```

9、编译安装 zabbix

```
./configure --sysconfdir=/etc/zabbix/ --enable-server --enable-agent --with-net-snmp --with-libcurl --with-mysql
make && make install
#此处指定 sysconfdir 配置文件的路径就在/etc/zabbix/目录下了，如果不指定默认在/usr/local/etc 下
```

10、Copy zabbixserver 端跟 agent 端的启动脚本，并设置执行权限

```
cp misc/init.d/tru64/zabbix_agentd /etc/init.d/
cp misc/init.d/tru64/zabbix_server /etc/init.d/
chmod +x /etc/init.d/zabbix_*
```

11、将 zabbix 的页面文件 copy 到指定目录(跟 apache 配置的相同即可)

```
mkdir /var/www/html/zabbix
cp -a    zabbix-2.2.2/frontends/php/* /var/www/html/zabbix/
chown -R    apache.apache /var/www/html/zabbix/
```

12、配置 php 文件，适应 zabbix 安装所需的参数

```
vim /etc/php.ini
date.timezone = Asia/Shanghai
max_execution_time = 300
max_input_time = 300
post_max_size = 32M
memory_limit = 128M
mbstring.func_overload = 2
```

13、配置 apache 文件，定义安装访问 zabbix 的虚拟主机

```
vim /etc/httpd/conf/httpd.conf
ServerName 127.0.0.1
<VirtualHost *:80>
```

```
 DocumentRoot   "/var/www/html"
 ServerName 192.168.239.130
</VirtualHost>
```

14、配置 zabbix server 端的文件，定义数据库的 IP、用户名、密码

```
vim /etc/zabbix/zabbix_server.conf
DBHost=192.168.239.130
DBName= zabbix
DBUser=zabbixuser
DBPassword=zabbixpass
StartPollers=30                         #开启多线程数，一般不要超过 30 个
StartTrappers=20                        #trapper 线程数
StartPingers=10                         #fping 线程数
StartDiscoverers=120
MaxHousekeeperDelete=5000
CacheSize=1024M                         #用来保存监控数据的缓存数，根据监控主机的数量适当调整
StartDBSyncers=8                        #数据库同步时间
HistoryCacheSize=1024M
TrendCacheSize=128M                     #总趋势缓存大小
HistoryTextCacheSize=512M
AlertScriptsPath=/etc/zabbix/alertscripts
LogSlowQueries=1000
```

15、启动 apache 服务跟 zabbix 服务

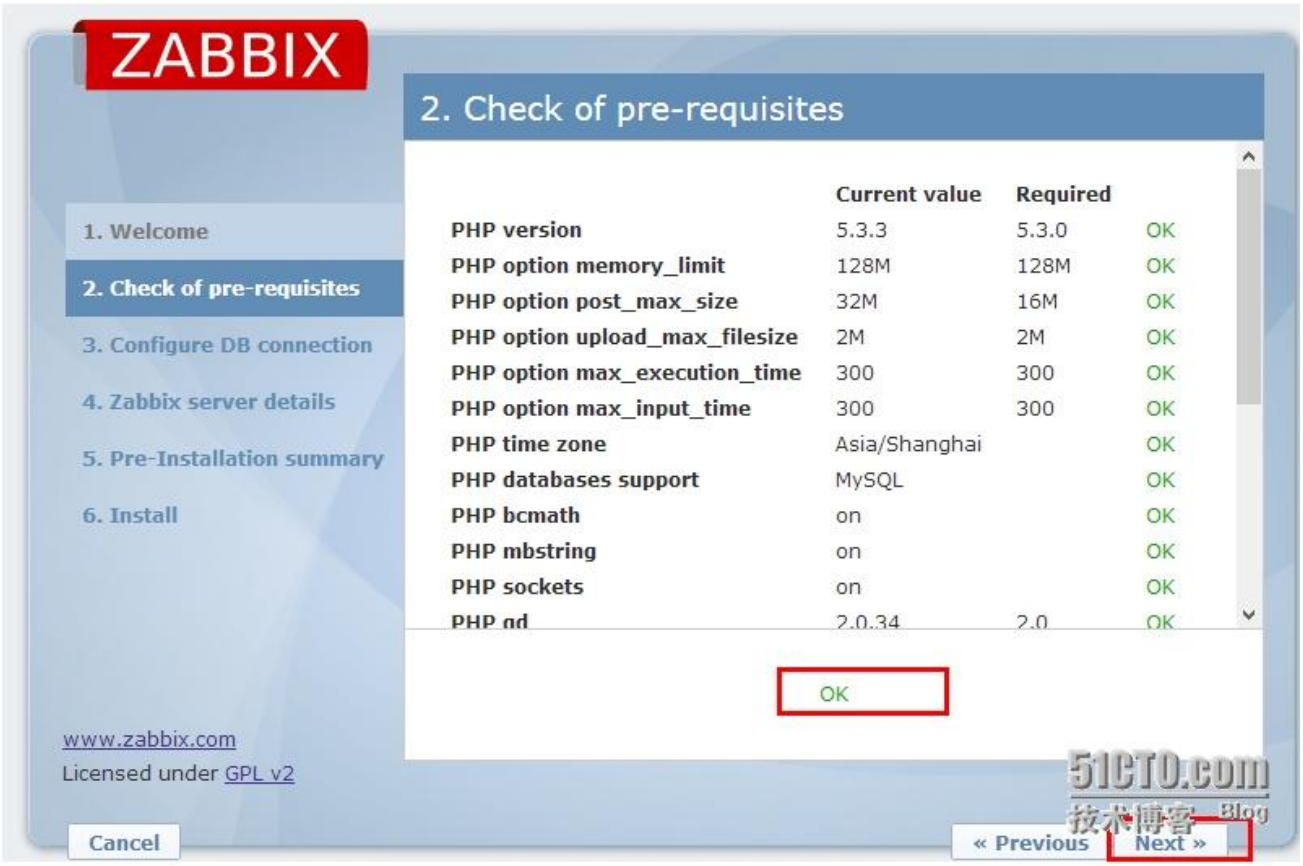
```
/etc/init.d/httpd start
/etc/init.d/zabbix_server start
```

16、访问安装界面按照界面提示一步一步的完成安装

```
http://192.168.239.130/zabbix/setup.php
```

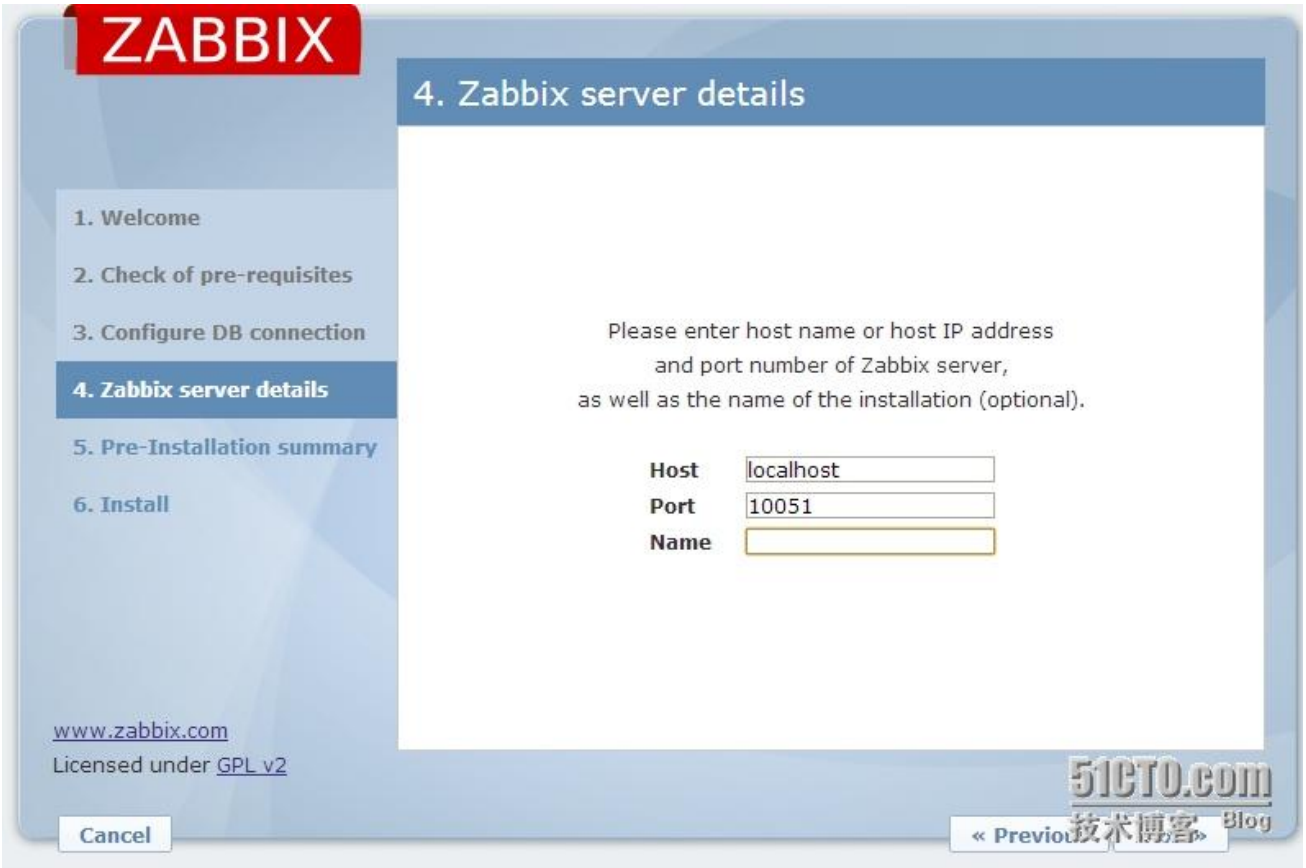
a) 进入安装界面点击 Next

b) 确保所有的监测项都是 OK，点击 Next



c) 填写 zabbix 数据库的用户名、密码、地址等信息 ,点击 Test connection,OK 后点击 Next

d) 填写zabbix服务器的信息,主机名 ,server程序监听的的端口 ,主机IP地址等 ,如果server跟 web 在一台服务器上保持默认即可，点击 Next



e) 确认前面几部填写的信息没有问题的话点击 Next

f) 检查 web 程序的 config 文件，如果没问题会显示 OK，直接点击 finish 即可完成安装(如果此处有问题一般是 zabbix 虚拟主机目录文件的权限问题，上面已经有设置的过程一般不会出现问题)

```
#至此，server端的安装完成
```

Agent 端

1、安装开发软件包

```
yum -y groupinstall "Development Tools"
yum -y install ntpdate
```

2、同步客户端时间，防止跟服务器端不一致，导致检测到不可用的监控数据

```
ntpdate pool.ntp.org
```

3、创建 zabbix 运行所需要的用户跟组

```
cd /usr/src/
tar xf zabbix-2.2.2.tar.gz
cd zabbix-2.2.2
./configure --sysconfdir=/etc/zabbix --enable-agent
make && make install
```

4、解压安装 zabbixagent 端

```
cd /usr/src/
tar xf zabbix-2.2.2.tar.gz
cd zabbix-2.2.2
./configure --sysconfdir=/etc/zabbix --enable-agent
make && make install
```

5、copy agent 端运行所需要的脚本

```
cp misc/init.d/tru64/zabbix_agentd /etc/init.d/
chmod +x /etc/init.d/zabbix_agentd
```

6、配置 agent 端配置文件

```
vim /etc/zabbix/zabbix_agentd.conf          #此处千万别写成了 zabbix_agent.conf,否则配置了不生效
Server=192.168.239.130                      #填写 Server 的 IP 地址
ServerActive=192.168.239.130                #修改为 Server 的 IP 地址
Hostname=Centos-03                          #填写本机的 HostName,注意 Server 端要能解析
UnsafeUserParameters=1                      #是否允许自定义的 key,1 为允许，0 为不允许
Include= etc/zabbix/zabbix_agentd.conf.d/#自定义的 agentd 配置文件(key)可以在这里面写；
```

7、启动 zabbix agent 端

```
/etc/init.d/zabbix_agentd start
```

解决 zabbix 中文乱码、汉化

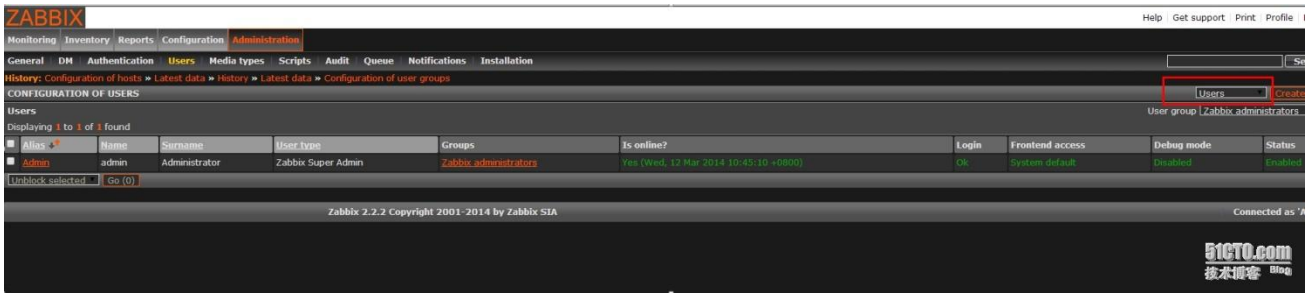
1、 在 windows 中找一个自己喜欢的字体或者去网上下载一个字体

2、将字体上传至/var/www/html/zabbix/fonts 目录下

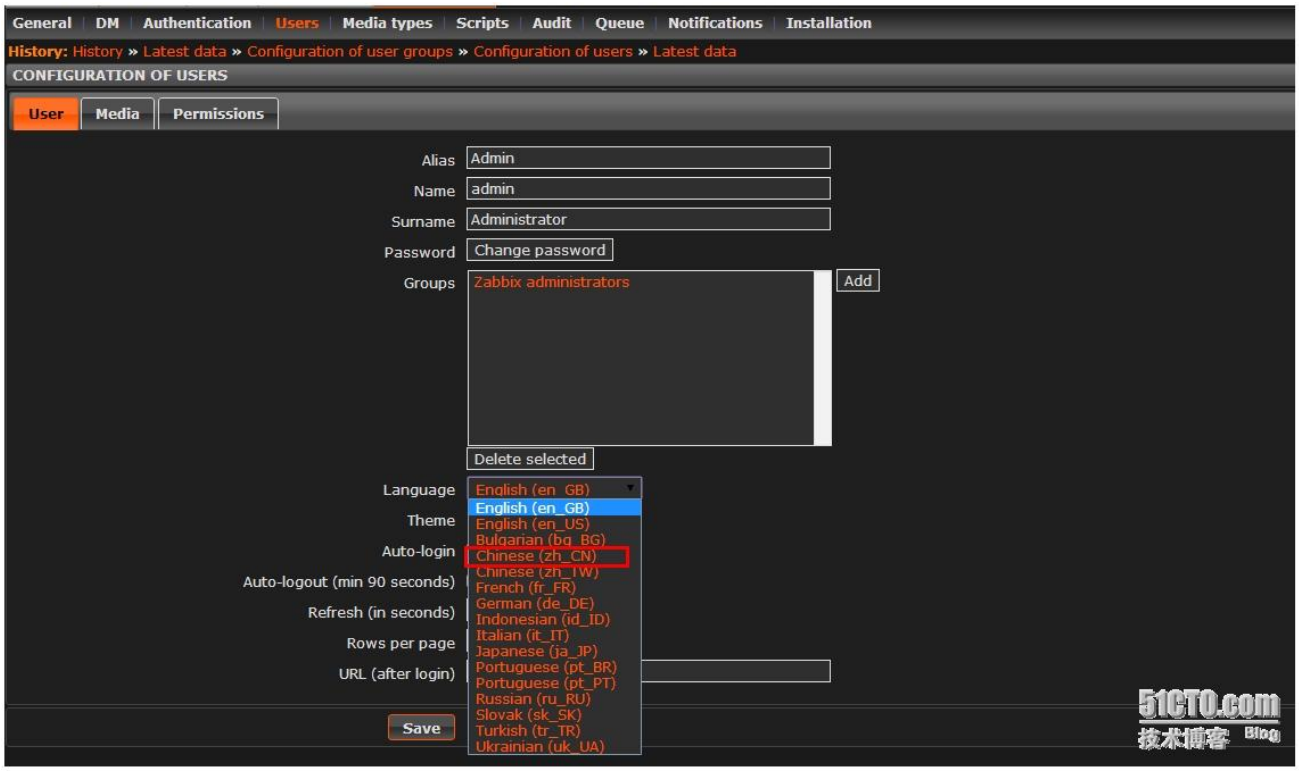
3、修改 zabbix 页面管理的中文字体设置

```
vim /var/www/html/zabbix/include/defines.inc.php          #修改以下两行
define('ZBX_FONT_NAME', 'simkai');
define('ZBX_GRAPH_FONT_NAME',  'simkai');
```

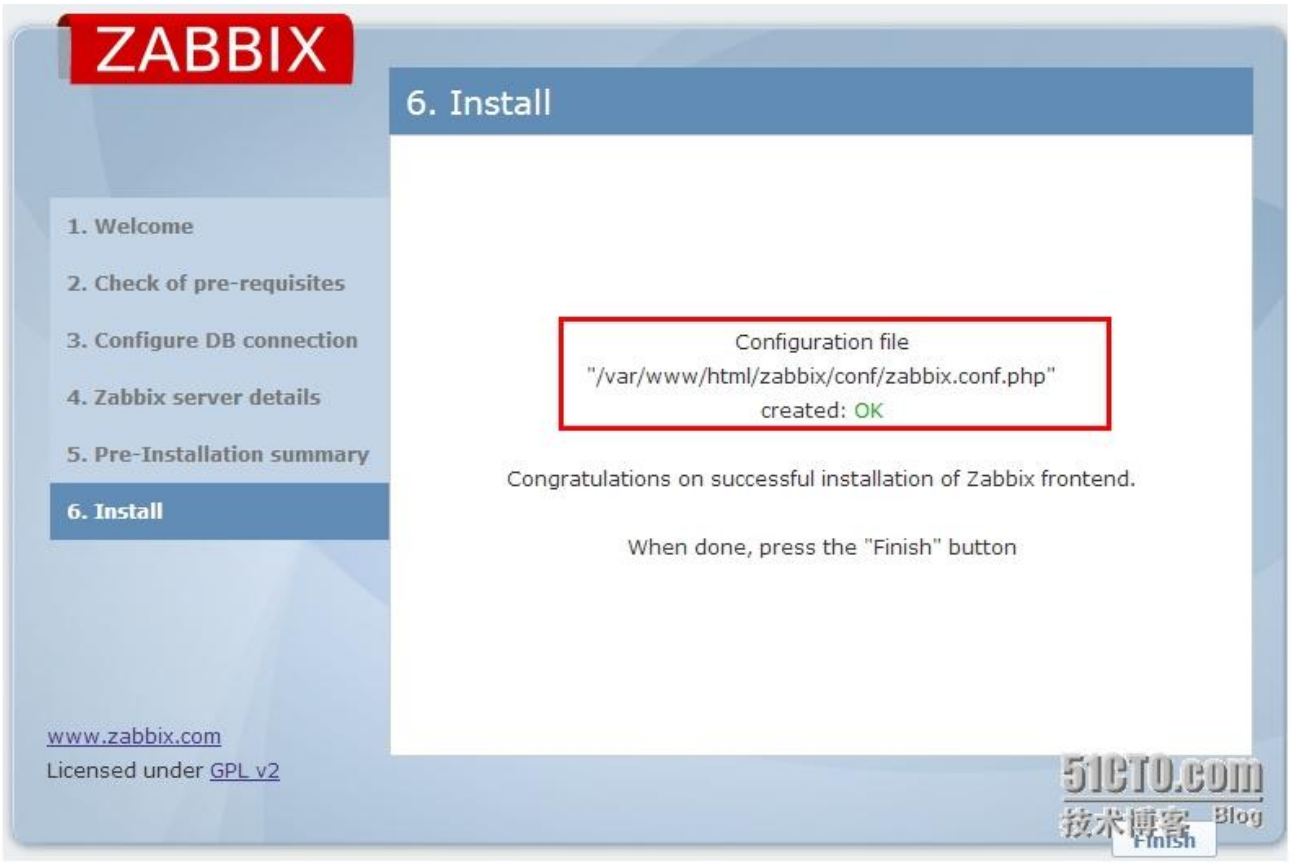
4、登陆页面设置相应用户的默认语言 Administrator---->Users(此处要保证显示的是用户，否则显示的都是用户组)



5、点击用户名进入用户信息编辑，默认语言选择简体中文，然后点击 save 保存



6、刷新页面



#如果因为手误，数据库密码或者用户名等的填写错了，后面会一直报错，我们可以手动的去编辑配置文件
vim /var/www/html/zabbix/conf/zabbix.conf.php 手动定义相关的参数即可

g) 进入登录界面点击登录，默认 admin zabbix


```
#已经成为中文，在有些地方还是会有英文的但是zabbix的汉化相对其他开源软件来说已经做的十分的好了！！！！！！.
```

Centos6.3 下单系统多 mysql 实例配置

作者：李震宇 来源：http://showerlee.blog.51cto.com/2047005/1378606

最近因为单位项目扩充,需要在原线上数据库服务器上加装一个 mysql 实例(实际上就是从新编译安装一个非 3306 的自定义端口,不同目录的 mysql),研究了一天,终于顺利搞定,这里把配置步骤发给大家,供大家学习使用.

注:本文档做了两个 MYSQL 实例,多个实例方法以此类推

LINUX 操作系统:centOS6.3 64bit(安装了系统默认开发包)
实例一:
MYSQL 版本:mysql-5.0.56
PORT:3306
系统目录:/usr/local/mysql3306

实例二:
MYSQL 版本:mysql-5.1.72
PORT:3307
系统目录:/usr/local/mysql3307

一.安装开发包(使用默认 CENTOS 更新源):
yum -y install wget gcc-c++ ncurses ncurses-devel cmake make perl bison openssl openssl-devel gcc* libxml2 libxml2-devel curl-devel libjpeg* libpng* freetype*

二.关闭 iptables 和 SELINUX
service iptables stop
setenforce 0
vi /etc/sysconfig/selinux

SELINUX=disabled

三.安装 mysql 数据库实例

1.下载编译包：

su -

mkdir ~/src

cd src

wget http://mysql.cdpa.nsysu.edu.tw/Downloads/MySQL-5.1/mysql-5.1.73.tar.gz

wget http://down1.chinaunix.net/distfiles/mysql-5.0.56.tar.gz

2.安装前的初始配置工作：

1).创建一个 Mysql 用户

useradd mysql

2).新建 mysql 下 data 和 log 子目录

mkdir /usr/local/mysql{3306,3307}/data

mkdir /usr/local/mysql{3306,3307}/log

3).修改目录的所属者以及所属组权限

chown -R mysql:mysql /usr/local/mysql{3306,3307}/data/

chown -R mysql:mysql /usr/local/mysql{3306,3307}/log/

chmod 750 /usr/local/mysql{3306,3307}/data

chmod 750 /usr/local/mysql{3306,3307}/log

4).创建 mysql 相关目录并配置权限

mkdir -p /usr/local/mysql{3306,3307}/etc

chown -R mysql.mysql /usr/local/mysql{3306,3307}/etc

mkdir -p /var/run/mysqld{3306,3307}

chown -R mysql.mysql /var/run/mysqld{3306,3307}

mkdir -p /var/lib/mysqld{3306,3307}

chown -R mysql.mysql /var/lib/mysqld{3306,3307}

cp /etc/my.cnf /usr/local/mysql{3306,3307}/etc

3.解包编译安装

编译实例一:

cd ~/src

tar -zxvf mysql-5.0.56.tar.gz

cd mysql-5.0.56

./configure --prefix=/usr/local/mysql3306 --with-mysqld-user=mysql
--sysconfdir=/usr/local/mysql3306/etc --localstatedir=/usr/local/mysql3306/data
--with-tcp-port=3306 -enable-assembler --with-mysqld-ldflags=-all-static --with-charset=utf8
--with-extra-charsets=gbk -with-extra-charsets=all --with-plugins=csv,innobase,myisam,heap
--with-unix-socket-path=/tmp/mysql3306.sock
make
make install
编译实例二:
cd ~/src
tar -zxvf mysql-5.1.71.tar.gz
cd mysql-5.1.71
./configure --prefix=/usr/local/mysql3307 --with-mysqld-user=mysql
--sysconfdir=/usr/local/mysql3307/etc --localstatedir=/usr/local/mysql3307/data
--with-tcp-port=3307 -enable-assembler --with-mysqld-ldflags=-all-static --with-charset=utf8
--with-extra-charsets=gbk -with-extra-charsets=all --with-plugins=csv,innobase,myisam,heap
--with-unix-socket-path=/tmp/mysql3307.sock
make
make install

4.编写 mysql 配置项：
实例一:
vi /usr/local/mysql3306/etc/my.cnf

[mysqld]
datadir=/usr/local/mysql3306/data
socket=/tmp/mysql3306.sock
user=mysql
port=3306
pid-file=/var/lib/mysqld3306/mysql.pid
Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0

max_connections= 16384

```
skip-name-resolve
skip-locking
key_buffer = 256M

max_allowed_packet = 32M
table_cache = 3072
thread_cache_size = 256
sort_buffer_size = 16M
read_buffer_size = 4M
read_rnd_buffer_size = 16M
net_buffer_length = 256M
thread_stack = 8M
query_cache_size = 128M
query_cache_limit = 2M
wait_timeout=7200
interactive_timeout=7200

#log
log-error=/usr/local/mysql3306/log/error.log
log=/usr/local/mysql3306/log/mysql.log
long_query_time=2
log-slow-queries= /usr/local/mysql3306/log/slowquery.log
log-bin= /usr/local/mysql3306/log/bin.log
expire_logs_days = 15
sync_binlog = 1
max_binlog_cache_size = 4294967295
local-infile=0

[mysqld_safe]
log-error=/var/log/mysqld3306.log
pid-file=/var/run/mysqld3306/mysqld.pid
-----------------------------------------------------
```

实例二:

vi /usr/local/mysql3307/etc/my.cnf

```
[mysqld]
datadir=/usr/local/mysql3307/data
socket=/tmp/mysql3307.sock
user=mysql
port=3307
pid-file=/var/lib/mysqld3307/mysql.pid
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0

max_connections= 16384
skip-name-resolve
skip-locking
key_buffer = 256M

max_allowed_packet = 32M
table_cache = 3072
thread_cache_size = 256
sort_buffer_size = 16M
read_buffer_size = 4M
read_rnd_buffer_size = 16M
net_buffer_length = 256M
thread_stack = 8M
query_cache_size = 128M
query_cache_limit = 2M
wait_timeout=7200
interactive_timeout=7200

#log
log-error=/usr/local/mysql3307/log/error.log
log=/usr/local/mysql3307/log/mysql.log
```

long_query_time=2
log-slow-queries= /usr/local/mysql3307/log/slowquery.log
log-bin= /usr/local/mysql3307/log/bin.log
expire_logs_days = 15
sync_binlog = 1
max_binlog_cache_size = 4294967295
local-infile=0

[mysqld_safe]
log-error=/var/log/mysqld3307.log
pid-file=/var/run/mysqld3307/mysqld.pid

5.将 mysql 的库文件路径加入系统的库文件搜索路径中
方法一：直接做软链接
ln -s /usr/local/mysql3306/lib/mysql /usr/lib/mysql
方法二：利用 ldconfig 导入系统库
echo "/usr/local/mysql3306/lib" >> /etc/ld.so.conf.d/mysql.conf
ldconfig

6.输出 mysql 的头文件到系统头文件
ln -s /usr/local/mysql3306/include/mysql /usr/include/mysql
注: 此处只需将一个 mysql 实例的库文件添加到系统库,无需多次添加

7.进入相应实例的安装路径,初始化各自配置脚本
实例一:
cd /usr/local/mysql3306
scripts/mysql_install_db --user=mysql --datadir=/usr/local/mysql3306/data
实例二:
cd /usr/local/mysql3307
scripts/mysql_install_db --user=mysql --datadir=/usr/local/mysql3307/data

8.复制 mysql 启动脚本到系统服务目录,并更改脚本配置

cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld3306

cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld3307

实例一:

vi /etc/init.d/mysqld3006

搜索如下行,红色标注的为添加的参数:

---

basedir=/usr/local/mysql3306

datadir=/usr/local/mysql3306/data

conf=/usr/local/mysql3306/etc/my.cnf

$bindir/mysqld_safe --defaults-file=$conf --datadir=$datadir --pid-file=$server_pid_file

$other_args >/dev/null 2>&1 &

实例二:

vi /etc/init.d/mysqld3007

搜索如下行,红色标注的为添加的参数:

---

basedir=/usr/local/mysql3307

datadir=/usr/local/mysql3307/data

conf=/usr/local/mysql3307/etc/my.cnf

$bindir/mysqld_safe --defaults-file=$conf --datadir=$datadir --pid-file=$server_pid_file

$other_args >/dev/null 2>&1 &

9.系统启动项相关配置

实例一:

chkconfig --add mysqld3306 #添加开机启动服务

chkconfig --level 35 mysqld3306 on #设置 mysql 启动

实例二:

chkconfig --add mysqld3307

chkconfig --level 35 mysqld3307 on

10.启动 mysql

实例一:

service mysqld3306 start

实例二:

service mysqld3307 start

11 添加 mysql 命令集到系统全局变量

注:如果系统之前未安装 mysql 客户端,可以将编译好的 mysql 命令集导入系统全局变量

以后就可以直接使用 mysql 命令集,而不需要使用绝对路径访问.

echo "PATH=$PATH:/usr/local/mysql3306/bin;export PATH" >> /etc/profile

source /etc/profile

12. 设置初始账户,并登陆后台:

实例一:

/usr/local/mysql3306/bin/mysqladmin -u root password 123456 #设置超级管理员密码

/usr/local/mysql3306/bin/mysql -P3306 -S/tmp/mysql3306.sock -uroot -p123456 #连接数据库

wKiom1Mn8ZiA_mTlAADxcsKLR5k148.jpg

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 565
Server version: 5.0.56-log Source distribution

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

实例二:

/usr/local/mysql3307/bin/mysqladmin -u root password 123456 #设置超级管理员密码

/usr/local/mysql3307/bin/mysql -P3307 -S/tmp/mysql3307.sock -uroot -p123456 #连接数据库

wKiom1Mn8mSB5C49AADmRTmPQ3g181.jpg

注: 因为加了 mysql 环境变量,以后系统后台可以直接使用 mysql 命令登录,这里使用绝对路径是为了规范操作

mysql -P3307 -S/tmp/mysql3307.sock -uroot -p123456

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 264
Server version: 5.1.72 Source distribution

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

13.端口测试:

wKioL1Mn8ujS5RnHAAF08bZNePs425.jpg

```
[root@sql src]# lsof -i:3306
COMMAND   PID  USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
mysqld  12949 mysql   15u  IPv4 15886102      0t0  TCP *:mysql (LISTEN)
mysqld  12949 mysql   30u  IPv4 15888601      0t0  TCP 192.168.10.20:mysql->192.
168.10.22:52321 (ESTABLISHED)
[root@sql src]# lsof -i:3307
COMMAND   PID  USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
mysqld  13457 mysql   10u  IPv4 15887680      0t0  TCP *:opsession-prxy (LISTEN)
mysqld  13457 mysql   29u  IPv4 15893790      0t0  TCP 192.168.10.20:opsession-p
rxy->192.168.10.22:50312 (ESTABLISHED)
mysqld  13457 mysql   30u  IPv4 15889175      0t0  TCP 192.168.10.20:opsession-p
rxy->192.168.10.22:49755 (ESTABLISHED)
[root@sql src]#
```

大功告成.....O(∩_∩)O~

CentOS6.5 linux 系统定制与封装快速实施脚本

作者：文斌 来源：http://lidongni.blog.51cto.com/2554605/1374457

在大型企业或者是 IDC 做运维的朋友经常会有这样子的需求，有些人可能还不是很了解 Linux 的二次封装有什么好处，这里简单说一下，就是当你要反复做一个操作的时候，比如说要部署 300 台 WEB 服务器，这时候你如果一台一台装，然后装完系统后再去配置这个 WEB 应用，那要花多少时间啊，是吧！但是如果你装好一台服务器，然后将这个系统进行二次封装，这样子你下次再要部署相同的 WEB 服务器的时候就只需要拿这个封装的系统去部署就 OK ,这是不是简单多了 ,这时候有人可能会怀疑会不会存在驱动问题，当然系统本身 OK 的 ,他是没有问题的 ,除非你是特殊的硬件。其实简单的理解就是一个 windows 的 ghost 封装差不多。如果你还是不是很明白，可能你需要多读几次这段文字。前期有测试过网上的一些教程，发现在 CentOS6.0 以上都不太好用了，由于时间的原因，我这里只是贴出封装的 shell，其实整个还需要有一个比较复杂的过程，你可能需要用心去了解一下下面这个 shell 的意思，有需要的朋友可以联系我。当然我经常会用它来封装像 openvpn、vsftpd、web 等服务器，让它能够实现全自动化快速部署，以提高我的工作效率，后期有机会将上次发布的 vsftpd 服务器的封装包发现来大家分享一下，大侠不要拍砖！头已经在流血。

```
#!/bin/bash
#web:www.lidongni.com
SYSTEM_DIR=/home/lidongni
SYSTEM_NAME=lidongni
mount_ISO (){
mkdir -p /home/lidongni/Packages
mkdir -p /home/source
mkdir -p /mnt/cdrom1
mkdir -p /mnt/cdrom2
echo "请确保光驱里面有 ISO 文件，且可以被挂载"
echo "请输入 Y/y 继续,任意键则退出运行"
read ret
[ ${ret} != "Y" -a ${ret} != "y" ] && exit 1
echo   "echo 默认情况，挂载的是/dev/cdrom 到/mnt/cdrom1"
mount /dev/cdrom   /mnt/cdrom1
mount /dev/cdrom1 /mnt/cdrom2
 ls /mnt/cdrom2 |grep Packages
if [ "$?" != 0 ]
then
echo "光盘挂载不成功，请手动重新挂载，或者尝试本地 ISO 挂载"
echo "请输入本地 ISO 的路径:"
read ret
ls ${ret}
[ "$?" != 0 ] && echo "本地 ISO 不存在或者路径错误，退出运行" && exit 1
[ "$?" == 0 ] && mount -o loop ${ret}   /mnt/cdcom1
fi
```

```
}
copy_ISO_file (){
#awk   '/安装/{print $2}' install.log | sed -e '/^$/d' -e 's/^ //g' | grep -v FINISHED | grep -v ":" > /home/source/
packges.list
mkdir -p /home/source
cp -p /root/packges.list /home/source/packges.list
mkdir -p /home/lidongni/Packages/conf/
cp -p /root/PS/* /home/lidongni/Packages/conf/
for packges in $(cat /home/source/packges.list)
do
cp /mnt/cdrom1/Packages/$packges*   /home/lidongni/Packages
[ $? != 0 ] && echo "copy   $packges is faied!"&& cp /mnt/cdrom2/Packages/$packges*
/home/lidongni/Packages
done
rsync -a --exclude=Packages --exclude=repodata /mnt/cdrom2/ /home/lidongni/
cp -p /root/initrd.img /home/lidongni/isolinux/initrd.img
mv /home/lidongni/CentOS_BuildTag /home/lidongni/lidongni_BuildTag
}
rebuild_repo_xml (){
cd   ${SYSTEM_DIR}
rm -fv ${SYSTEM_DIR}/repodata/*
mkdir -p ${SYSTEM_DIR}/repodata/
cp -fv /mnt/cdrom2/repodata/*-c6-x86_64-comps.xml ${SYSTEM_DIR}/repodata/comps.xml
createrepo -u "media://$(head -1 .discinfo)" -g ${SYSTEM_DIR}/repodata/comps.xml ${SYSTEM_DIR}/
find ${SYSTEM_DIR}/repodata/ -type f -name TRANS.TBL | xargs /bin/rm -fv
}
isolinux_cfg (){
cat > ${SYSTEM_DIR}/isolinux/isolinux.cfg <<EOF
default auto
timeout 600
#default vesamenu.c32
#prompt 1
display boot.msg
menu background splash.jpg
menu title Welcome to lidongni 6.4!
menu color border 0 #ffffffff #00000000
menu color sel 7 #ffffffff #ff000000
menu color title 0 #ffffffff #00000000
menu color tabmsg 0 #ffffffff #00000000
menu color unsel 0 #ffffffff #00000000
menu color hotsel 0 #ff000000 #ffffffff
menu color hotkey 7 #ffffffff #ff000000
menu color scrollbar 0 #ffffffff #00000000
label auto
menu label ^Auto Install system
```

```
menu default
kernel vmlinuz
append ks=cdrom:/isolinux/ks.cfg initrd=initrd.img
label linux
menu label ^Install or upgrade an existing system
kernel vmlinuz
append initrd=initrd.img
label vesa
        menu label Install system with ^basic video driver
       kernel vmlinuz
        append initrd=initrd.img xdriver=vesa nomodeset
       label rescue
        menu label ^Rescue installed system
       kernel vmlinuz
   append initrd=initrd.img rescue
label local
 menu label Boot from ^local drive
 localboot 0xffff
label memtest86
   menu label ^Memory test
 kernel memtest
 append -
EOF
}
ks_file(){
cat > ${SYSTEM_DIR}/isolinux/ks.cfg <<EOF
# Kickstart file automatically generated by anaconda.
#lidongni
#date 2013-07-20
#version=V6.4
install
text
cdrom
lang zh_CN.UTF-8
keyboard us
skipx
network --device eth0 --bootproto static --ip 192.168.50.1 --netmask 255.255.255.0 --gateway 192.168.50.254
--hostname lidongni-FTP
rootpw --iscrypted /gvPu5dZM6eis
firewall --service=ssh
authconfig --enableshadow --passalgo=sha512
selinux --enforcing
timezone --utc Asia/Shanghai
bootloader --location=mbr --driveorder=sda --append=" rhgb crashkernel=auto quiet"
# The following is the partition information you requested
```

```
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
#clearpart --none
clearpart --all --initlabel
zerombr yes
part /boot --fstype=ext4 --size=100
part swap --size=4000
part / --fstype=ext4 --size=10000
part /opt --fstype=ext4 --grow --size=1000
reboot
#repo --name="lidongni"  --baseurl=cdrom:sr0 --cost=100
%packages  --nobase
@base
%packages
@chinese-support
@core
glibc-2.12-1.107.el6.i686
lrzsz
%post --nochroot
cp /mnt/source/Packages/conf/settings.sh /mnt/sysimage/tmp/settings.sh
mv /mnt/sysimage/etc/vsftpd/vsftpd.conf /mnt/sysimage/etc/vsftpd/vsftpd.conf.back
mv /mnt/sysimage/etc/pam.d/vsftpd /mnt/sysimage/etc/pam.d/vsftpd.back
cp /mnt/source/Packages/conf/vsftpd.conf /mnt/sysimage/etc/vsftpd/vsftpd.conf
cp /mnt/source/Packages/conf/ftpmgt.exe /mnt/sysimage/etc/vsftpd/ftpmgt.exe
cp /mnt/source/Packages/conf/vsftpd /mnt/sysimage/etc/pam.d/vsftpd
cp /mnt/source/Packages/conf/moban /mnt/sysimage/etc/vsftpd/moban
cp /mnt/source/Packages/conf/xianzhi /mnt/sysimage/etc/vsftpd/xianzhi
chmod 750 /mnt/sysimage/tmp/settings.sh
%post
/tmp/settings.sh
EOF
}
build_new_ISO (){
cd ${SYSTEM_DIR}
mkisofs -o ${SYSTEM_NAME}.iso -b isolinux/isolinux.bin -c isolinux/boot.cat      -no-emul-boot   -boot-load-size
4   -boot-info-table   -R   -J   -v   -V   lidongni   -T ${SYSTEM_DIR}
     }
main (){
mount_ISO
copy_ISO_file
isolinux_cfg
ks_file
rebuild_repo_xml
build_new_ISO
```

```
}
main
```

通过 vmware tools 来为克隆出来的虚拟机配置 IP 地址

作者：Leon 之　来源：http://leontam.blog.51cto.com/8150854/1379914

Windows 和 Linux 配置 IP 地址都很简单，但前提是你能直接访问它们。
如果是一个刚完成克隆的虚拟机，如何配置 IP 地址呢？我们不能总是人工去做，但没配 IP 之前，系统自身的一切接口都不能用，只能从 vmware tools 上想办法了。上一篇里提到 vmware tools 可以在没网络的情况下传输文件，其实它还可以在没网络的情况下登录到操作系统中执行指定命令。

以下是通过 pysphere 来调 vmware tools，进入虚拟机中执行命令的代码，目前可以在 ubuntu/RedHat/Windows2003 上使用。其实只要定制相关的脚本或命令，vmwaretools 只是起一个通道的功能。

```
def ChangeVM_IP(vm,vm_os,vm_ip,vm_netmask,vm_gateway,vm_main_dns,vm_passwd=None):

if CommonDefinition.simulation:
    return True
if vm_os=='ubuntu':
    cmd_path='/bin/echo'
    #echo ces | sudo -S /opt/ecloud/reconfig_ubuntu_network.sh'

cmd_args=[vm_passwd,'|','sudo','-S','/opt/ecloud/reconfig_ubuntu_network.sh',vm_ip,vm_netmask,vm_gateway,vm_main_dns]
    try:
        pid=vm.start_process(cmd_path,args=cmd_args)
        time.sleep(10)
        return True
    except Exception,e:
        msg='Error in executing change ip command for %s.' % vm.get_property('name')
        myloger.error(msg)
        myloger.debug(trace_back())
        return False

if vm_os=='rhel5':
    cmd_path='/opt/ecloud/reconfig_network.sh'
    cmd_args=[vm_ip,vm_netmask,vm_gateway,vm_main_dns]
    try:
        pid=vm.start_process(cmd_path,args=cmd_args)
        time.sleep(10)
        return True
    except Exception,e:
        msg='Error in executing change ip command for %s.' % vm.get_property('name')
        myloger.error(msg)
```

```
            myloger.debug(trace_back())
            return False
if vm_os=='windows2003':
    myloger.debug('config windows ip:')
    windows2003_eth_connection_name=VMware_CommonDefinition.windows2003_eth_connection_name
    cmd_path='C:\\WINDOWS\\system32\\netsh.exe'
    cmd_args_str='interface ip set address name="%s" source=static addr=%s mask=%s gateway=%s
gwmetric=1' % (windows2003_eth_connection_name,vm_ip,vm_netmask,vm_gateway)
    cmd_args=cmd_args_str.split(' ')
    try:
        pid=vm.start_process(cmd_path,args=cmd_args)
        time.sleep(30)
        """
        real_vm_ip=str(vm.get_property('ip_address'))
        myloger.debug('real ip: %s' % real_vm_ip)
        if vm_ip!=str(vm.get_property('ip_address')):
            pid=vm.start_process(cmd_path,args=cmd_args)
            time.sleep(60)
            if vm_ip!=str(vm.get_property('ip_address')):
                myloger.error('VM %s ip config error. Can not change ip to %s' %
(vm.get_property('name'),vm_ip))
                return False

        """
    except Exception,e:
        msg='Error in executing change ip command for %s.' % vm.get_property('name')
        myloger.error(msg)
        myloger.debug(trace_back())
        return False

    #config dns
    cmd_args_str='interface ip set dns name="%s" source=static addr=%s register=PRIMARY' %
(windows2003_eth_connection_name,vm_main_dns)
    cmd_args=cmd_args_str.split(' ')
    try:
        time.sleep(5)
        pid=vm.start_process(cmd_path,args=cmd_args)
        time.sleep(10)
        return True
    except Exception,e:
        msg='Error in executing change dns command for %s.' % vm.get_property('name')
        myloger.error(msg)
        myloger.debug(trace_back())
        return False
```

相关脚本：

ubuntu:

```
echo "">/etc/network/interfaces
sed -i -e "$ i\auto lo \n\
iface lo inet loopback\n\
auto eth0\n\
iface eth0 inet static\n\
address $1\n\
netmask $2\n\
gateway $3\n\
dns-nameservers $4" /etc/network/interfaces
/etc/init.d/networking restart
```

RedHat5:

```
#!/bin/bash
#for redhat5
echo "">/etc/sysconfig/network-scripts/ifcfg-eth0
sed -i -e "$ i\DEVICE=eth0 \n\
BOOTPROTO=static\n\
NM_CONTROLLED=yes\n\
IPADDR=$1\n\
NETMASK=$2\n\
GATEWAY=$3\n\
ONBOOT=yes\n\
TYPE=Ethernet" /etc/sysconfig/network-scripts/ifcfg-eth0
sed -i -e "s/\(nameserver \)\S*/\1$4/" /etc/resolv.conf
service network restart
```

impala 集成 kerberos 问题一例

作者：菜菜光 来源：http://caiguangguang.blog.51cto.com/1652935/1381323

最近在折腾 hadoop+kerberos，由于线上使用的组件比较多，遇到不少问题，记录下来，碰到同样问题的同学可以参考下。

在 hdfs+mapred+kerberos 运行正常后，开始尝试集成 impala.
其中 statestore 的参数：

```
export IMPALA_STATE_STORE_ARGS=${IMPALA_STATE_STORE_ARGS:- -log_dir=${IMPALA_LOG_DIR} \
-state_store_port=${IMPALA_STATE_STORE_PORT} -kerberos_reinit_interval=60
-principal=impala/xxxxxx@KERBEROS_HADOOP -keytab_file=/etc/impala/conf.dist/impala.keytab}
```

impala-server 的参数：

```
export IMPALA_SERVER_ARGS=${IMPALA_SERVER_ARGS:- -log_dir=${IMPALA_LOG_DIR} \
-state_store_port=${IMPALA_STATE_STORE_PORT} -    use_statestore
-state_store_host=${IMPALA_STATE_STORE_HOST} \
 -be_port=${IMPALA_BACKEND_PORT}
-statestore_subscriber_timeout_seconds=${STATESTORE_SUBSCRIBER_TIMEOUT_SECONDS} -mem_limit=50% \
-kerberos_reinit_interval=60 -principal=impala/xxxxx@KERBEROS_HADOOP
-keytab_file=/etc/impala/conf.dist/impala.keytab}
```

启动 statestore 没有异常，因为在 impala 1.1.1 版本中，statestore 只是做一个监控 impala-server 进程的作用，不涉及和 hadoop 的通信，而在启动 impala-server 时，发现进程运行一段时间之后就会 crash，通过设置 impala 的日志级别 export GLOG_v=3，可以在日志中观察到下面的错误：

```
E0305 17:29:06.696974 12551 UserGroupInformation.java:1411] PriviledgedActionException
as:impala/datanode@KERBEROS_HADOOP (auth:KERBEROS)
cause:java.io.IOException: Couldn't setup connection for
impala/gd6g12s103-hadooptest-datanode.idc.vipshop.com@KERBEROS_HADOOP to
hdfs/namenode@KERBEROS_HADOOP
E0305 17:29:06.699252 12551 impala-server.cc:339] Could not read the HDFS root directory at hdfs://bipcluster.
Error was:
Failed on local exception: java.io.IOException: Couldn't setup connection for
impala/gdatanode@KERBEROS_HADOOP to
hdfs/namenode@KERBEROS_HADOOP; Host Details : local host is: "datanode/ip";
destination host is: "namenode":8020;
E0305 17:29:06.699296 12551 impala-server.cc:341] Aborting   Impala Server startup due to improper
configuration
```

可以看到确实再用 kerbers 做验证登陆，但是在 datanode 和 namenode 通信时出现错误，因为线上

用了 namenode 的 ha，在日志中发现有 ha 的报错，因为怀疑是 ha 的问题，在关闭 ha 后，问题仍然存在。

日志中还有 tgt 相关的报错：

```
Caused by: javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism level: Failed to find any Kerberos tgt)]
```

在 datanode 端，运行 hadoop fs -ls 的命令时，报错。通过 export HADOOP_ROOT_LOGGER=DEBUG,console 设置 hadoop 命令的日志级别，发现也是同样报了 tgt 相关的错误。

在通过 klist 查看 tgt 的 cache，发现 tgt 竟然过期了，而且不能进行 kinit -R.

```
klist
Ticket cache: FILE:/tmp/krb5cc_501
Default principal: hdfs/namenode@KERBEROS_HADOOP
Valid starting     Expires            Service principal
03/11/14 18:45:52  03/12/14 18:45:52  krbtgt/KERBEROS_HADOOP@KERBEROS_HADOOP
    renew until 03/11/14 18:45:56
```

这是由于 renew expires 导致，kerberos 中有两个时间比较重要：

max_list,tgt 的有效时间，max_renewable_life ,renew 的时间，在 max_renewable_life 时间内，过期的 tgt 可以 renew，如果时间超过 max_renewable_life 就不能 renew 了。。

查看线上的设置：

```
max_life = 25h
max_renewable_life = 4w
```

而实际 renew 的最大时间却是 4s（03/11/14 18:45:56-03/11/14 18:45:52），看来 w 不是 week 的意思。。不知道算不算 bug，修正下，改成 30d，重新 kinit，就正常了。。

后面如果报 Kerberos: Couldn't find mech GSSAPI 说明是 cyrus-sasl-gssapi 的相关包没有安装。

启动正常后验证：

```
impala-shell -i   ip -k   -s impala
Starting Impala Shell in secure mode (using Kerberos)
[10.19.111.106:21000] > use cdnlog;
Query: use cdnlog
[10.19.111.106:21000] > select count(1) from dd_log;
Query: select count(1) from dd_log
Query finished, fetching results ...
+----------+
| count(1) |
+----------+
| 5000000  |
```

```
+----------+
```

可以看到已经正常跑了，自己对 kerberos 的了解还是太少了，在解决 kerberos 的相关问题的时候，第一步就应该用 klist 验证。

一个真实的案例———HPUX 调整 LUN 大小识别更改

作者：田文丰 来源：http://crazy123.blog.51cto.com/1029610/1380076

#uname -a
HP-UX xxxx B.11.31 U ia64

磁盘阵列通常允许调整 LUN 的大小,如果增加 LUN 的大小,请执行以下步骤将附加空间合并到卷组中：
1、按照阵列说明增加 LUN 的大小。
2、运行 vgmodify 检测任何物理卷大小更改。还将报告卷组能否使用所有空间。
3、如果 vgmodify 报告，每个物理卷的最大物理盘区数 (max_pe) 太小，无法容纳新增的空间，请使用带 -t 和 -n 选项的 vgmodify 确定 max_pe 的新值，如“修改卷组参数”
4、按照新的设置运行带 -r 选项的 vgmodify 检查这些值。
5、停用卷组。
6、提交 max_pe 的任何新值，运行不带 -r 选项的 vgmodify 更新物理卷信息。
7、激活卷组。运行 vgdisplay 和 pvdisplay 命令验证增加的空间是否可用。

详细实施步骤:
一步：备份 VG 配置信息
#vgcfgbackup -f /home/vg01_bak vg01

二步：选择合适的 max_pv 与 max_pe

做这一步的原因是 VG 默认的 Max PE per PV=5960，而 PE Size (Mbytes) =64，并不能满足 LUN 扩展后 PV 的容量需求
需要调整 VG 属性值。

#vgmodify -v -r vg01 #查看 vg01 当前的 VG 配置信息

#vgmodify -t vg01 #查看可选的卷组配置信息，会看到一张 max_pv，max_pe 与 Disk_size 的对照表

#vgmodify -t -v -n vg01 #与上条命令类似，会生成更大磁盘容量的 max_pv，max_pe 与 Disk_size 的对照表

找到一个合适的值，主要是确定 max_pv 数与每 PV 支持的 max_pe 数组合能满足扩展后物理卷要求
从列表可以看出当 max_pv<14 都可以满足需求，这里选择 max_pv=12，max_pe=16124

三步：修改 VG 的属性
#vgmodify -p 12 -e 16124 -r vg01 #先预览下调整后的效果

#fuser -cu /oracle/data #列出正在使用/oracle/data 的进程信息
#fuser -ku /oracle/data #kill 掉正使用/oracle/data 的进程

#umount /oracle/data #卸载文件系统（不是必须的）

#vgchange -a -n vg01 #关闭 vg01

#vgmodify -p 12 -e 16124 vg01 #修改 vg01 属性值

#vgchange -a -y vg01 #激活 vg01

四步：扩展文件系统
#extendfs -L 819200 /dev/vg01/lvo1 #扩展文件系统到 819200M

五步：挂载文件系统
#mount /dev/vg01/lvo1 /oracle/data

六步：查看调整后的 VG
#vgdisplay -v vg01

实施过程的几个疑点
问一：调整 LUN 大小后 HUUX 能否立即识别，怎么确认？
回答：LUN 调整后，HPUX 操作系统可以立即识别，可使用以下命令检查：diskinfo /dev/rdisk/disk9

问二：调整 LUN 大小后，对应的 PV 是否会自动增加容量？
回答：主要看 PV 所在卷组属性，如果 Max PE per PV*PE Size > 扩展后 PV 则可以识别到，反之则识别不到，需要调整 VG 属性值

将第二步与第四步的一些输出信息贴出来做个参考：

```
# vgmodify -v -r vg01
Volume Group configuration for /dev/vg01 has been saved in /etc/lvmconf/vg01.conf
Current Volume Group settings:
                                                Max     LV        255
                                                Max     PV         12
                                        Max PE per      PV      16124
                                    PE Size     (Mbytes)         64
                                   VGRA Size     (Kbytes)       1600
An update to the Volume Group is NOT required
```

```
      123                 65535               4194304
      124                 65532               4194112
      125                 65020               4161344
      126                 64508               4128576
      127                 63996               4095808
      128                 63484               4063040
      129                 62972               4030272
      130                 62460               3997504
      --                  -----               -------
      252                 32252               2064192
      254                 31996               2047808
      255                 31740               2031424
# vgdisplay -v vg01
--- Volume groups ---
VG Name                       /dev/vg01
VG Write Access               read/write
VG Status                     available
Max LV                        255
Cur LV                        1
Open LV                       1
Max PV                        12
Cur PV                        2
Act PV                        2
Max PE per PV                 16124
VGDA                          4
PE Size (Mbytes)              64
Total PE                      20839
Alloc PE                      12800
Free PE                       8039
Total PVG                     0
Total Spare PVs               0
Total Spare PVs in use        0
VG Version                    1.0
VG Max Size                   12093g
VG Max Extents                193488
   --- Logical volumes ---
   LV Name                        /dev/vg01/lvol1
```

一个引号导致 1 个小时网站打不开

> 作者：贺春旸　来源：http://hcymysql.blog.51cto.com/5223301/1369853

咱们就说下这个例子，提醒广大开发在写 SQL 的时候一定要仔细！

当时情况是这样的，一个慢 SQL 把数据库 CPU 连接数跑满，由于并发压力大，CPU 空闲瞬时为 0，过一会机器被 HANG 死，连接不上。

因涉及公司隐私问题，我这里用测试表代替，咱们主要看看是怎么引起的。

表结构：

```
mysql> desc sbtest;
+-------+------------------+------+-----+---------+-------+
| Field | Type             | Null | Key | Default | Extra |
+-------+------------------+------+-----+---------+-------+
| id    | int(11)          | NO   | PRI | 0       |       |
| k     | int(10) unsigned | NO   | MUL | 0       |       |
| c     | char(120)        | NO   |     |         |       |
| pad   | char(60)         | NO   |     |         |       |
+-------+------------------+------+-----+---------+-------+
4 rows in set (0.22 sec)
```

现在开始了，开发写了这么一条 SQL 语句：

```
select * from sbtest where id in ('1','2','111111111111');
```

（注：11 个 1）

各位，你们觉得这条 SQL 有问题吗？很简单，开发也认为这么简单的 SQL 不用给 DBA 去审核，但往往阴沟里翻船，死在了自认为简单的 SQL 里，那么我们执行一下，看看执行计划。

```
mysql> explain select * from sbtest where id in ('1','2','11111111111');
+----+-------------+--------+------+---------------+------+---------+------+---------+-------------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows    | Extra       |
+----+-------------+--------+------+---------------+------+---------+------+---------+-------------+
|  1 | SIMPLE      | sbtest | ALL  | PRIMARY       | NULL | NULL    | NULL | 1000109 | Using where |
+----+-------------+--------+------+---------------+------+---------+------+---------+-------------+
1 row in set (0.00 sec)
```

id 是主键，却没有用到索引，全表扫描。这是为毛？

溢出了。int 最大宽度是 11 位，而我刚才输入了 11 个 1，溢出了，如果换成 10 个 1，再来看看效果。

```
mysql> explain select * from sbtest where id in ('1','2','1111111111');
+----+-------------+--------+-------+---------------+---------+---------+------+------+-------------+
| id | select_type | table  | type  | possible_keys | key     | key_len | ref  | rows | Extra       |
+----+-------------+--------+-------+---------------+---------+---------+------+------+-------------+
|  1 | SIMPLE      | sbtest | range | PRIMARY       | PRIMARY | 4       | NULL |    3 | Using where |
+----+-------------+--------+-------+---------------+---------+---------+------+------+-------------+
1 row in set (0.00 sec)
```

已经用到索引，只需扫描 3 行就出结果。

下面再看看这两条 SQL 的执行时间：

```
mysql> select * from sbtest where id in ('1','2','11111111111');
+----+---+---+-------------------------------------------------------------+
| id | k | c | pad                                                         |
+----+---+---+-------------------------------------------------------------+
|  1 | 0 |   | qqqqqqqqqqwwwwwwwwwweeeeeeeeeerrrrrrrrrrttttttttttt |
|  2 | 0 |   | qqqqqqqqqqwwwwwwwwwweeeeeeeeeerrrrrrrrrrttttttttttt |
+----+---+---+-------------------------------------------------------------+
2 rows in set (4.39 sec)
```

插入 11 个 1，耗时 4.39 秒

```
mysql> select * from sbtest where id in ('1','2','1111111111');
+----+---+---+-------------------------------------------------------------+
| id | k | c | pad                                                         |
+----+---+---+-------------------------------------------------------------+
|  1 | 0 |   | qqqqqqqqqqwwwwwwwwwweeeeeeeeeerrrrrrrrrrttttttttttt |
|  2 | 0 |   | qqqqqqqqqqwwwwwwwwwweeeeeeeeeerrrrrrrrrrttttttttttt |
+----+---+---+-------------------------------------------------------------+
2 rows in set (0.10 sec)
```

插入 10 个 1，耗时 0.10 秒

结论：

为了避免这种问题的出现，int 数值整形不要加''引号，如果是 varchar 字符串类型，要加上''引号。血的教训！请各位开发注意！数据库是业务的核心，不能想当然，自己写的痛快就直接跑现网，造成的损失是极大的。

JQuery 高性能最佳实践

作者：安大叔 来源：http://andashu.blog.51cto.com/8673810/1375329

【使用最佳选择器】

使用 JQuery 时，你可以使用多种选择器，选择同一个元素，各种方法之间的性能是不一样的，有时候差异会特别大。 通常比较常用的选择器有以下几个：

1. ID 选择器 $("#id")
2. 标签选择器 $("td")
3. 类选择器 $(".target")
4. 属性选择器 $("td[target='target']")
5. 伪类选择器 $("td:hidden")

根据经验，我们应该知道这 5 种选择器的性能是依次下降的，我们不妨来做个测试，看看他们的性能到底有多大差异：

测试 html 片段：

```
<table width="98%" cellspacing="1" cellpadding="0" border="0" style="table-layout:fixed" id="mytable">
    <tr>
      <td bgcolor="#aaaaaa" align="center" class="target" target="target" style="display:none;" id="target-td">e</td>
    </tr>
</table>
```

测试结果

测试方案：对每个脚本执行 1w 次，统计 3 次运行结果的平均值

方案	IE6	IE7	IE8	IE9	chrome	firefox	opera	safari
$("#mytable td.target")	5150	5630	780	293	69	148	31	102
$("#mytable .target")	5320	5780	940	297	61	141	32	101
$("#mytable").find("td.target")	4840	5000	1250	387	95	205	73	157
$("#mytable").find(".target")	5000	5150	1400	226	49	130	60	64
$("#mytable td[target=target]")	16410	17660	940	406	89	166	35	120
$("#mytable td:hidden")	25000	26720	23750	3638	632	1123	3434	569
$("#target-td")	630	620	310	62	9	28	12	18
$(".target")	10310	10790	940	207	36	181	47	44
document.getElementById("target-td")	150	150	160	6	1	1	5	2

结论

原生方法是最快的方法，如果可能，尽量选择用原生

性能顺序：ID 选择器 > 标签选择器 > 类选择器 > 属性选择器 > 伪类选择器

ID(getElementById)、标签选择器(getElementsByTagName)都有原生的方法对应，所以很快；类选择器在除了 IE5-IE8 之外的主流浏览器几乎都有原生方法(getElementsByClassName)

为了兼顾 IE6、7、8 的性能，避免使用全局的类选择器；

属性和伪类选择器非常慢，如非必要，尽量少使用伪类选择器和属性选择器

最佳实践

为模块中操作最频繁的元素和根元素设置 id，然后缓存；

对没有 id 的元素检索，尽量采用路径最短的祖先元素 id 作为默认的搜索范围，并采用纯类选择器或者标签选择器；

尽量避免复杂的选择器

【避免执行全局查找】

```
var blank=$(".blank");//length=0
```

A：

```
 blank.slideUp();
```

B:

```
blank.length && blank.slideUp();
```

测试结果

- 测试说明：1w 次执行耗时，单位毫秒/ms,统计三次运行的平均值

方案	IE6	IE7	IE8	IE9	chrome	firefox	opera	safari
A	6110	5610	1344	488	103	194	108	155
B	0	0	0	0	0	0	0	0

结论

应该避免对空对象进行操作；

【采用样式表，避免多次调整样式】

对一个对象应用多个样式，最好采用样式表的方式，避免多次应用。

var obj=$("#obj");

- A:

```
obj.css("width",200);

obj.css("height",200);

obj.css("background":"#eee");
```

- B:

```
obj.attr("style","width:200px;height:200px;background:#eee;");
```

- C:

```
.css-operation{width:200px;height:200px;background:#eee;}


obj.addClass("css-operation")
```

测试结果

- 测试说明：1w 次执行耗时，单位毫秒/ms,统计三次运行的平均值

方案	IE6	IE7	IE8	IE9	chrome	firefox	opera	safari
A	2594	2486	1500	501	163	222	190	191
B	1000	953	547	190	79	28	15	86
C	843	672	407	111	21	17	16	31

结论

1. 性能排序：C>B>A
2. 样式和 JS 分离的方案性能最佳，适用于要同时设置多个样式的场景；
3. 如果只应用单个样式，简单起见可以考虑采用方案 A
4. 如果应多若干个样式，而且不愿意新建一个 css class，可以采用 B；

【避免使用匿名函数】

大量的使用匿名函数会对程序的调试、维护以及通过第三方软件来做性能测试增加难度，因此应该尽量避免大量的使用匿名函数

```
obj.click(function(){

  //do something...

})
```

=>>

```
var clickHandler=function(){

  //do something...

}

obj.click(clickHandler)
```

【大循环采用更高效的遍历方式】

JQuery 提供了$.fn.each()和$.each()两个方法来实现对集合的遍历，除此之外，还可以采用 JS 原生的 for 循环、while 等来实现迭代，应该了解一下他们之间的性能差异：

```
var list=ul.find("li"),e;
```

- A:

```
var i=list.length;
while(i--){
    e=$(list[i])
}
```

- B:

```
list.each(function(){
   e=$(this);
});
```

- C:

```
$.each(list,function(){
   e=$(this);
});
```

测试结果

- 测试说明：1w 次执行耗时，单位毫秒/ms,统计三次运行的平均值

方案	IE6	IE7	IE8	IE9	chrome	firefox	opera	safari
A	172	219	157	30	3	5	4	6
B	219	234	203	41	4	6	5	8
C	219	234	187	52	3	4	5	7

结论

1. 总体上来说 A>C>B
2. 方案 A 有大约 25%的性能提升，但是不稳定；
3. 在 IE 浏览器下 B 方案和 C 方案性能相当，A 方案有比较绝对的优势；
4. Chrome、firefox 下 A 方案的性能不稳定；

最佳实践

1. 追求极致性能，用方案 A；
2. 循环数量少的话，建议使用方案 C，比较稳定；

【优先使用原生属性】

很多常用的属性，比如 id、name 等都被浏览器原生实现，在 JQuery 中我们有时会用$(this).attr("id")的方式来获取 id，这种方法的效率相比原生属性的获取效率而言，非常慢。

```
$.each(list,function(){
    //A
    var id=$(this).attr("id");
    //B
    var id=this.id;
})
```

测试结果

- 测试说明：10w 次执行耗时，单位毫秒/ms,统计三次运行的平均值

方案	IE6	IE7	IE8	IE9	chrome	firefox	opera	safari
A	6880	7030	4220	1188	157	244	133	135
B	310	310	150	27	4	5	17	3

结论

1. 使用原生的 API，可以极大的提高性能

最佳实践

1. 对于 id 等常用的属性，用原生的属性，不要通过 attr 去获取；

【使用事件委托】

经常会遇到给一个列表中所有元素添加点击事件的业务场景，传统的做法是得到这个列表的 JQuery 对象:$("li"),然后添加 click 事件：

```
$("li").click(function(){})
```

这种方法的在列表数量比较大的时候会有严重的性能问题，应该值得关注。JQuery 在很早的版本中已经引入了事件委托机制，可以很大程度的降低添加事件监听的消耗和内存的消耗。

对 1w 条记录的列表进行测试：

- A:

```
var list=$("li");//length>1

   list.click(function(){

})
```

- B:

```
$("ul").delegate("li","click",function(){})
```

测试结果

- 测试说明：对 1w 个<li>标签进行 click 事件添加的耗时，单位毫秒/ms,统计三次运行的平均值

方案	IE6	IE7	IE8	IE9	chrome	firefox	opera	safari
A	2156	2172	1922	312	103	173	141	117
B	0	0	0	0	0	0	0	0

结论

1. 委托的性能优势是非常绝对的；

最佳实践

1. 对于需要同时给两个以上的同类型元素添加事件时，用方案 B 来代替 A

【缓存查找的中间结果】

```
$(".list-item").show();
```

```
$(".list-item").hide();
```

=>

```
var listItem=$(".list-item");

listItem.show();

listItem.hide();
```

【减少 DOM 操作，尽量批量更新】

Dom 操作是浏览器操作中最为耗时的操作之一，JQuery 中提供了 append、appendTo、prepend、prependTo、after、before、insertAfter、wrap 等操作 dom 的实用方法，频繁使用这些方法可能会引起性能问题，一个提高性能的实践原则就是“尽可能少的使用它们”。 如果一定要用到，也尽可能的采用合并、批量操作来减少 dom 的操作消耗。

【使用$.data 而不是$.fn.data】

```
$(elem).data(key,value);

$.data(elem,key,value);
```

后者比前者快近 10 倍

【可能的话，使用最新版本的 JQuery】

新版本总会对性能进行改进，还会提供一些非常好用的工具，如果可以的话，应该尽量选用最新的版本；

【jQuery html 性能大坑】

jQuery 的 html 方法的作用是为 dom 元素设置 innerHTML，分析 html 的源代码（1.8.3）

```
if ( elem.nodeType === 1 ) {

    jQuery.cleanData( elem.getElementsByTagName( "*" ) );

    elem.innerHTML = value;
```

```
}
```

在设置 dom 的 innerHTML 之前，会执行 jQuery.cleanData，这个方法会对 dom 元素做一些 clean 的处理，如 removeEvent，删除缓存等。
以两百行的列表为例，在 ff 浏览器中，该方法会执行大约 5ms 到 8ms。即当 dom 元素为空时和 dom 元素中有两百行数据时，执行 html 方法，后者会比前者多运行 5ms 到 8ms。

坑点

cleanData 方法在 jQueryUI 中也会定义，且会重写$.cleanData，增加一些额外的操作，性能会受到影响。
还是以两百行的列表为例，在 ff 浏览器中，该方法会执行大约 60ms 到 70ms。即当 dom 元素为空时和 dom 元素中有两百行数据时，执行 html 方法，后者会比前者多运行 60ms 到 70ms。

解决方案

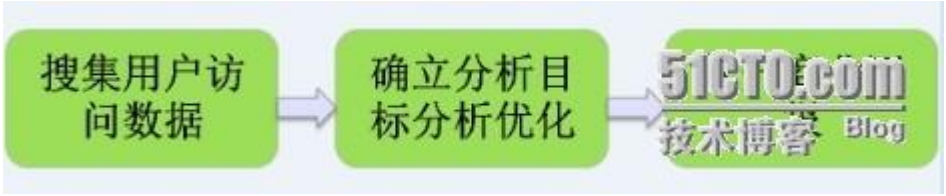
1. 采用原生的 dom.innerHTML
2.在执行 html()方法之前，先执行 remove()方法

前端日志分析

> 作者：安大叔　来源：http://andashu.blog.51cto.com/8673810/1375317

前端日志分析介绍

- 前端日志分析是通过搜集访客访问网站的行为数据，然后在这些用户日志数据的基础上通过定量和定性分析，来改善用户的浏览体验及网站性能，最终提升商业回报的过程，通常，前端日志分析遵循以下步骤：

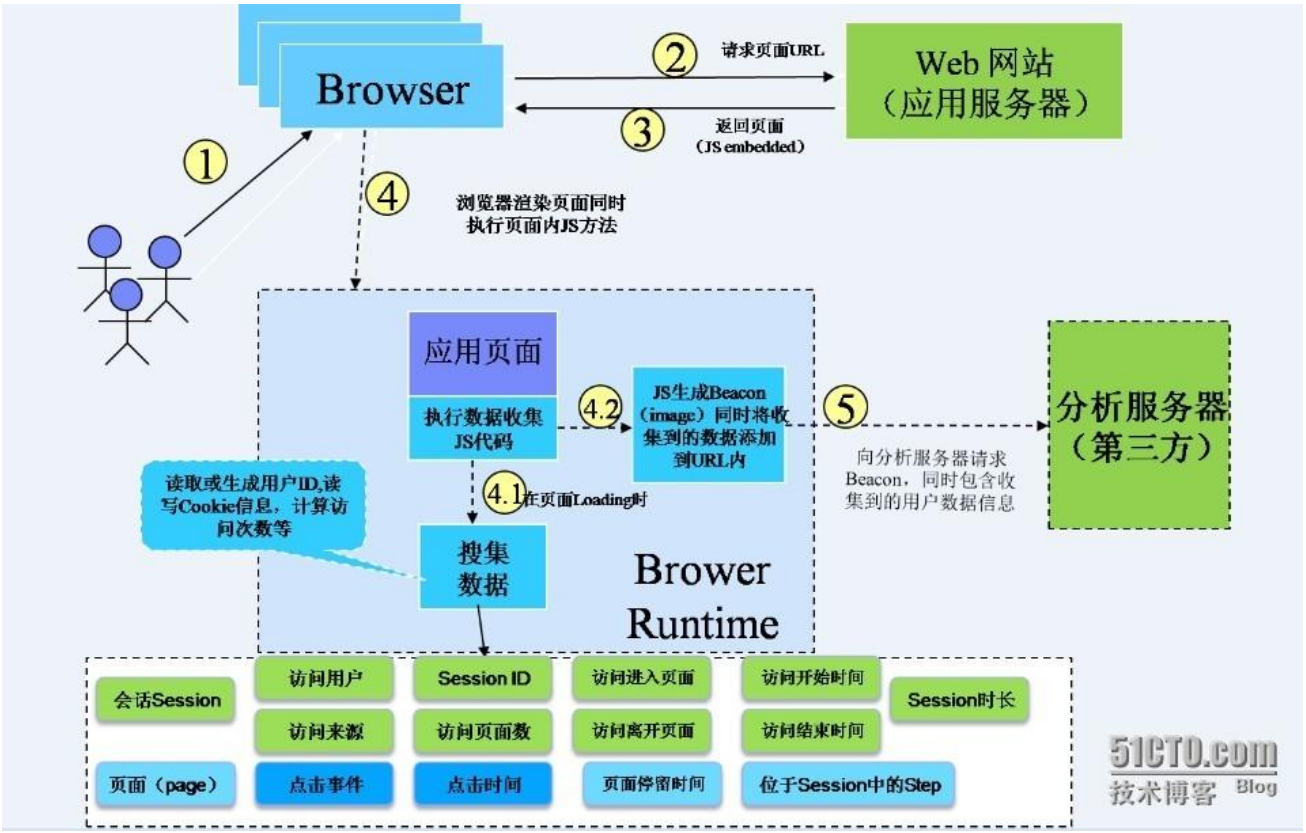


前端日志分析对性能测试的意义：

- 能确定性能指标，包括系统的并发量、响应时间、要测试的功能等
- 通过各项指标的变化幅度，确定系统性能健康度
- 容量规划的参考

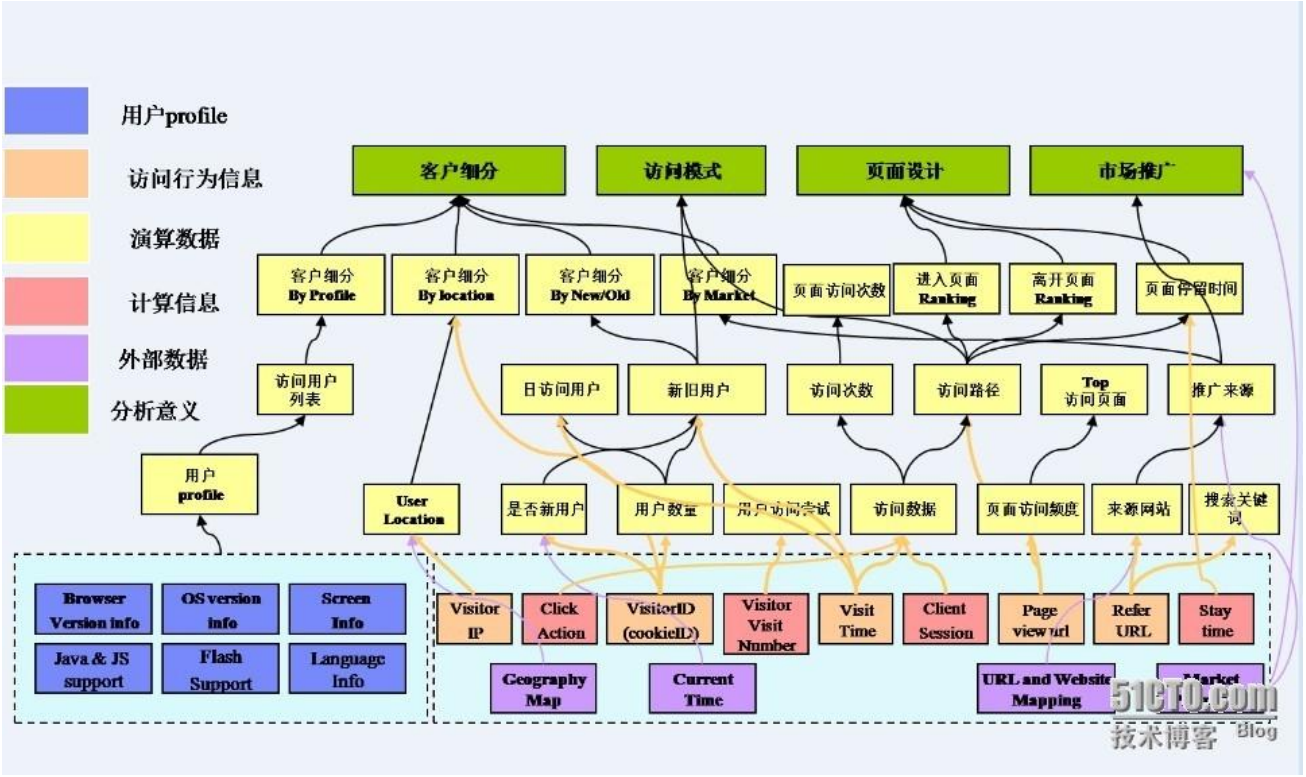
前端日志如何搜集

- 用户日志数据搜集原理如下图：



前端日志数据各部分意义

- 可搜集的各部分访问数据及意义：



前端日志数据分析目标

- 一般前端分析从以下五个角度进行

1. 流量分析
    1. **访问次数（PV,UV）**
    2. **流量来源（Referer）**
    3. **搜索流量与直接流量**
    4. **停留时间与页面尝试等**

2. 受众分析
    1. **语言与位置分布**
    2. **新旧用户分布**
    3. **操作系统与浏览器分布**
    4. **访问频次与时间分布**
    5. **流失率与活跃度分布**
    6. **用户价值分析等**

3. 页面分析

1. **网站内容访问情况**
2. **目标退出页面**
3. **页面访问速度**
4. **页面操作跟踪与热力图等**

4. 转化分析
   1. **目标设定与分析**
   2. **电子商务交易分析**
   3. **漏斗转化分析等**

5. 其它
   1. **A/B 测试**
   2. **ROI 分析等**

前端日志记录的字段设计

- 当前实现的前端日志中记录的字段如下表所示：

字段	取值说明
sy	window.location.hostname
ag	agentUserId || Null
ad	adminUserId || Null
aco	AgentCompanyId || Null
ac	CURRENT_P4P_USER_ID
an	window.navigator.appName
av	window.navigator.appVersion
pf	window.navigator.platform
url	当前监控url
s1	时间戳：发送请求
s2	时间戳：服务器响应
s3	时间戳：前端完成渲染
err	用户请求结果情况的标记字段 [见下表]

值(类型：字符串)	对应的情况
00	正常
01	未登录
02	需要验证码
03	禁止
04	出错 或 系统繁忙

构建故障分析平台采用 python 实现抓包分析数据包

作者：芮峰云 来源：http://rfyiamcool.blog.51cto.com/1030776/1374484

前言：

同事今天和我说，他现在的任务在做一个头疼的问题，说时尚了点，就是用自动化解脱心碎的运维杂事，他这边刚入职，貌似是带领一帮小弟解决别人搞不定的问题，但是有些业务部够单纯，把事直接抛给我同事这边。。。 很无敌吧。

所以计划做一个自动化平台，可以去问题端去抓数据，然后分析数据包，入库，邮件通知。 这个是自动化完成的。

我这里就说下，我的解决思路和开发思路：

工具：

pcap dpkt saltsatck mysql tornado tcpdump

pcap 是用来抓包

dpkt 是用来解析数据包的

celery 异步任务

实现两大功能，用户他自己抓包，然后上传到页面上，然后我后端解析后，返回结果。
用户在平台上提交问题服务器的 ip，并选定测试类型，我这里会到服务端跑用 python 的 pcap 抓包并分析结果，把结果上报到平台。

关于自动抓包分析，以前和同事做过处理 dns 攻击的，方案流程和第三方的工具和我上述是一样的。遇到攻击，会分析 dns 的攻击的特征，然后再黑洞系统注射特征码禁止。

需要注意的是，在抓数据的时候，可能会产生堵塞，尤其是 pcap，dpkt 这东西，需要在后台自己的玩。这个时候就需要用 celery 把抓包分析包的任务放在后台执行。 你要是觉得 subprocess 合理的话，也可以用用 subprocess 的 pipe 的，但是个人觉得也是个办法 ，更简单的方法是用 tcpdump -w 写到一个文件里面，然后用 dpkt 去解析，这样的话，也不用 pcap 去解析啦。

安装是相当的简单，不管是 centos 和 ubuntu 都已经有默认的源了，我这里用的是 ubuntu 的开发机跑测试：

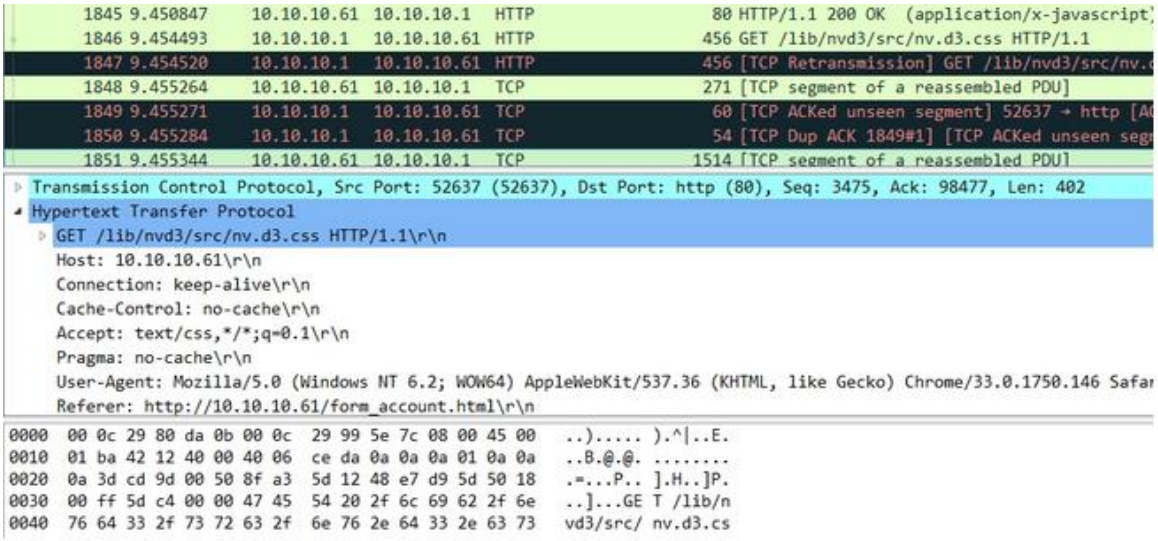
```
apt-get install python-libpcap
apt-get install libpcap-dev
apt-get install python-dpkt
pip install pypcap
```

原文：http://rfyiamcool.blog.51cto.com/1030776/1374484

下面是抓取 http 数据包的写法，大家可以慢慢取值，慢慢分拆数据。

```
#coding:utf-8
import pcap
import dpkt
import sys
aaa='a'
bbb='b'
pc=pcap.pcap()      #注，参数可为网卡名，如 eth0
pc.setfilter('tcp port 80')      #设置监听过滤器
for ptime,pdata in pc:      #ptime 为收到时间，pdata 为收到数据
p=dpkt.ethernet.Ethernet(pdata)
if p.data.__class__.__name__=='IP':
    ip='%d.%d.%d.%d'%tuple(map(ord,list(p.data.dst)))
#           print ip
    if p.data.data.__class__.__name__=='TCP':
        if p.data.data.dport==80:
            sStr1 = p.data.data.data
            sStr2 = 'Host: '
            sStr3 = 'Connection'
            sStr4 = 'GET /'
            sStr5 = ' HTTP/1.1'
            nPos = sStr1.find(sStr3)
            nPosa = sStr1.find(sStr5)
            for n in range(sStr1.find(sStr2)+6,nPos-1):
                aaa=sStr1[sStr1.find(sStr2)+6:n]
            for n in range(sStr1.find(sStr4)+4,nPosa+1):
                bbb=sStr1[sStr1.find(sStr4)+4:n]
            ccc=aaa+bbb
            print ccc
```

用 wireshark 瞅瞅：



原文：http://rfyiamcool.blog.51cto.com/1030776/1374484

我前两天在做 lvs 操作平台的时候，额外加了一个针对 vrrp 的分析数据模块，大家可以举一反三在改改！

```
#coding:utf-8
#xiaorui.cc
#http://rfyiamcool.blog.51cto.com/
import pcap
import dpkt
import binascii
import struct
a=pcap.pcap()
a.setfilter('vrrp')   # 可以是'tcp' 'udp' 'port 80'等过滤用的
for i,j in a:
tem=dpkt.ethernet.Ethernet(j)
print ("11111%s %x",i,tem)
src='%d.%d.%d.%d' % tuple(map(ord,list(tem.data.src)))
dst='%d.%d.%d.%d' % tuple(map(ord,list(tem.data.dst)))
print tem.data.tos
print tem.data.sum
print tem.data.len
print tem.data.ttl
print tem.data.id
#    print tem.data.data
print tem.data.data.priority
print src
print dst
```

通过获取的数据，可以得知对端的 vrrp 情况，比如 tos src dst vrrp 主信息 ！

记我真实的一段维护任务：程序查询慢到最快也需要 15 秒？

作者：shyy8712872 来源：http://shuyangyang.blog.51cto.com/1685768/1380328

程序报错：**开始的由于系统缓冲区空间不足或队列已满问题解决办法**

一般有 2 点原因：

a)系统内存不足，情况表现为空闲数低于 200 以下，系统句柄数大的可怕，达到 10 几 W

b)TCP 连接数不够，严重的可能导致数据库连接失败，项目部同事之前也说过这样的情况，就是此问题导致的，前提是物理内存/虚拟内存設定值都正常的情况下增加 TCP 连接数（可能是你留作种的原因,所以 tcp 的端口(UserPort)请求已经达到你 pc 上本地设置的界限(MaxUserPort)，默认的一般比较小，正广和的这台机器可能需要活动的 TCP 数量太多，所以需要设置大点），可以试着修改此键值，方法如下：

启动注册表编辑器，在注册表中，找到以下子项，然后单击 $参数(Parameters 翻译过来就是>>参数<<的意思) HKEY_LOCAL_MACHINESYSTEMCurrentControlSetServicesTcpipParameters

在编辑菜单中，单击新建，然后添加以下注册表项：

值名称: MaxUserPort

值类型： 双字节

值数据： 65534

有效范围： 5000-65534 （十进制）

默认值： 0x1388 （5000 十进制）

退出注册表编辑器，然后重新启动计算机。

2、针对一般的数据库，具有以下几点优化法则：

- 1、减少数据访问（减少硬盘访问，这个就是我们程序的事了，项目由于是老的项目，还是 ASP.NET 编写的，考虑到改源码起来麻烦，所以后面我经过仔细分析增加了索引）
- 2、返回更少的数据（减少网络传输或磁盘访问）
- 3、减少交互次数
- 4、减少服务器 CPU 及内存开销
- 5、利用更多的资源（增加资源）

一般处理这样的情况，

- 一、修改 SQL 语句，由于项目太老，更改源码麻烦，所以我采用了增加索引，建立索引的优点：（1.大大加快数据的检索速度; 2.创建唯一性索引，保证数据库表中每一行数据的唯一性; 3.加速表和表之间的连接; 4.在使用分组和排序子句进行数据检索时，可以显著减少查询中分组和排序的时间）

- 二、增加索引（聚集索引和唯一索引），在聚集索引中，表中行的物理顺序与键值的逻辑（索引）顺序相同。一个表只能包含一个聚集索引。 如果某索引不是聚集索引，则表中行的物理顺序与键值的逻辑顺序不匹配。与非聚集索引相比，聚集索引通常提供更快的数据访问速度。唯一索引，针对经常查询的字段，客户那边告诉我，有时间、工号和分机号。我主要对这三个经常的字段进行了增加索引。
- 建立索引的语法：CREATE 索引名称 ON 表名（需要创建索引的字段）；

经过这样的调优，速度最慢也可以 1 秒就查询出来了。好了，以上就是问题的解决方法和我个人的一些经验分享。

Hive 动态分区导致的 Jobtracker Hang

作者：MIKE 老毕 来源：http://boylook.blog.51cto.com/7934327/1380981

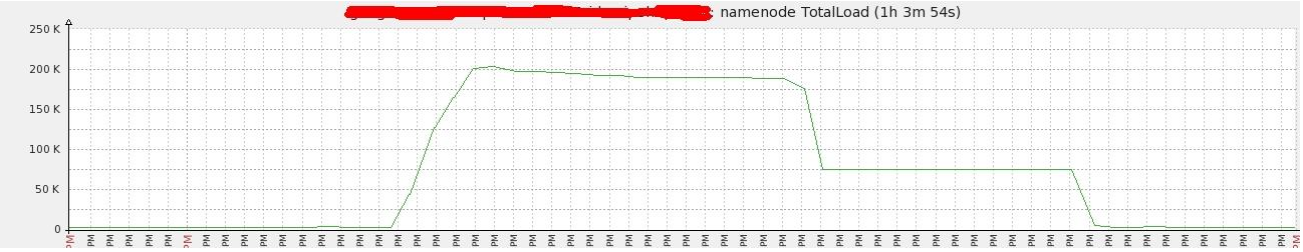
昨天下午有 20 多分钟 Hadoop 平台无法跑 Hive，Jobtracker 的页面也打不开，hadoop job –list 也 hang 住没有响应，过了 10 分钟后恢复了，查看 gc 日志发现 Jobtracker 没有进行 full gc，查看这段时间的 Job 日志发现一个可疑的 Hive SQL: Insert into table t(dt) as select xxx,dt from txx，是一个用了动态分区的查询.这个查询和 Jobtracker Hang 住有什么关系呢？

熟悉 Jobtracker 的都知道，在进行 Job 初始化时 EagerTaskInitializationListener 会锁住 JobInProgress 然后进行 InitTask,细节请各位查看代码 ,这里有一步就是需要向 hdfs 写入初始数据并 flush，而 Fairscheduler 的 Update Thread 在更新资源池的资源时是在持有 JobTracker 和 Fairscheduler 的独占锁然后再去计算每个资源池的资源情况 ,而计算 running_map/running_reduce 的时候要去获取相应的 JobInProgress 锁，各位读者可能不明白，我为啥要讲这块呢，问题就出现在这里.

Hive 在处理动态分区的时候，主要经历这么几个步骤 tablescan->filesink->movetask

在进行 filesink 的时候是根据记录来处理的，会起 N（part）个 record writer 然后开始处理动态分区字段 ,即这里的 dt ,如果 dt 是连续的那么打开一个 block 开始写 ,否则关闭当前 block ,打开新 dir 的 block 继续写，这里如果 dt 是不连续的出现并且记录数量巨大的情况下会产生大量的文件，导致 hdfs 的负载标高，和当时的 hdfs 的监控是匹配的：

当时的集群负载：



当时产生的文件数：

进而导致 JobInProgress 被锁住，从而 JobTracker 被锁住，导致 JobTracker Hang 住了！

那怎么解决呢？利用 distributeby dt 把相同的 dt 排列到一起再进行 filesink 就不会造成大量的小文件产生了.

使用 golang 的 http 模块构建 redis 读写查 api

作者： 芮峰云 来源：http://rfyiamcool.blog.51cto.com/1030776/1380754

前言：

这两天试着用 golang 做一些高性能的 api，不想把压力到聚合在平台的接口上。平台因为要做很多耗时间的操作，uwsgi 下会出现少许错误，找了一圈不知道如何解决该问题。 暂时先绕道而行，先拿简单的接口来做测试，慢慢的把复杂的操作也迁移到 golang 上。

话说以前高性能的接口，我用的最多的方案还是 nginx lua 的组合，超强，大家可以看看我以前写的 nginx lua 的文章，各方面没得说。只是这段时间正在看 golang 的，就试着用 golang 实现 redis 的 api，先来个简单的试试手。

里面引用的是 golang 自带的 http 模块，redis 是自己 down 的。

golang redis 的配置方法：

```
cd $GOPATH/src
git clone git://github.com/alphazero/Go-Redis.git redis
cd redis
go install
```

先简单说下，golang 对于 redis 的操作方法：

```
//xiaorui.cc
package main
import (
"os";
"bufio";
"log";
"fmt";
"redis";
)
/*
hello world, redis style.
*/
func main () {
    // create the client.  Here we are using a synchronous client.
// Using the default ConnectionSpec, we are specifying the client to connect
// to db 13 (e.g. SELECT 13), and a password of go-redis (e.g. AUTH go-redis)
spec := redis.DefaultSpec().Db(13).Password("go-redis");
client, e := redis.NewSynchClientWithSpec (spec);
if e != nil { log.Println ("failed to create the client", e); return }
key := "examples/hello/user.name";
```

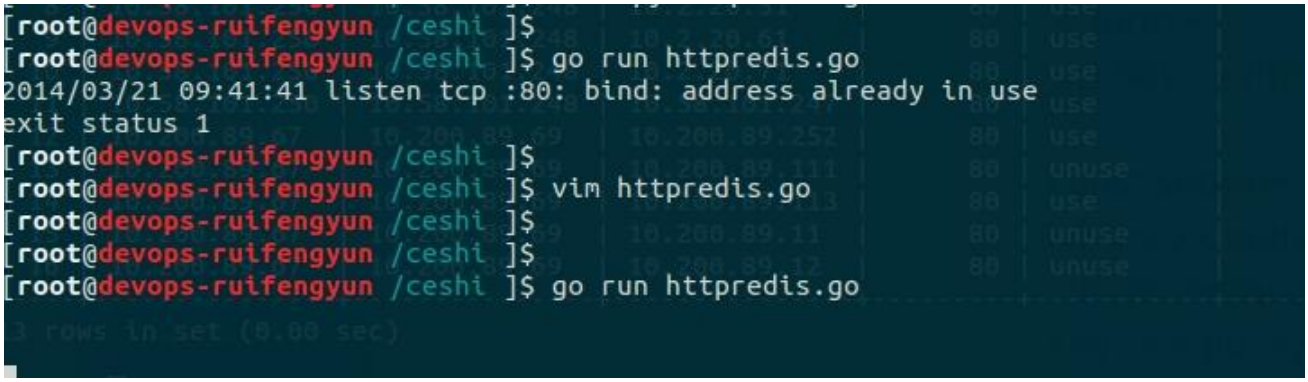
```
value, e := client.Get(key);
if e!= nil { log.Println ("error on Get", e); return }
if value == nil {
	fmt.Printf("\nHello, don't believe we've met before!\nYour name? ");
	reader:= bufio.NewReader(os.Stdin);
	user, _ := reader.ReadString(byte('\n'));
	if len(user) > 1 {
		user = user[0:len(user)-1];
		value = []byte(user);
		client.Set(key, value);
	} else {
		fmt.Printf ("vafanculo!\n");
		return;
	}
}
fmt.Printf ("Hey, ciao %s!\n", fmt.Sprintf("%s", value));
}
```

我写的实例，大家看懂了后，完全可以做更多的扩展。

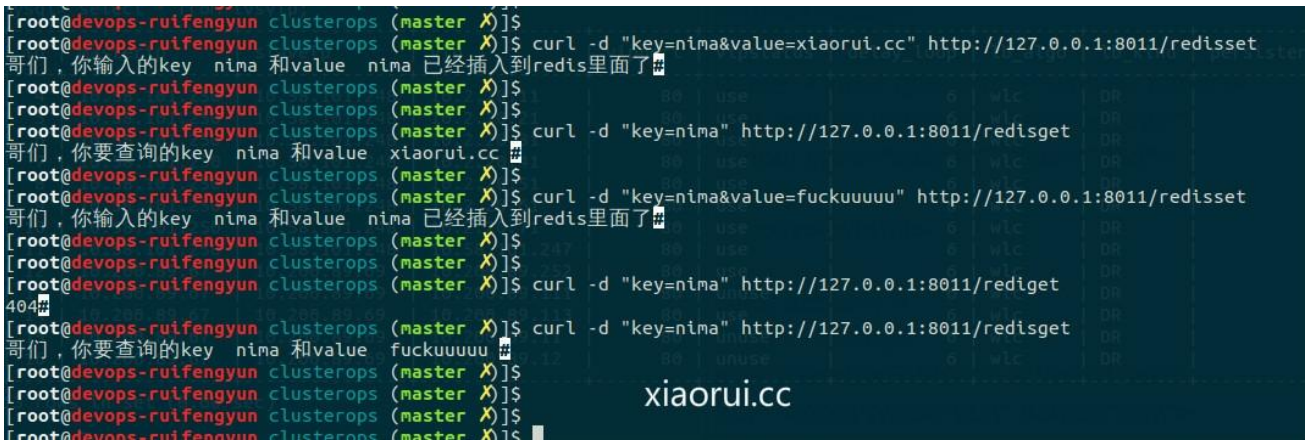
其实 golang 自带的 http 很有 mvc 的感觉 ,三者做了一些分离 ,很像 python 里面的 web.py tornado。。。

测试结果：

服务端的启动



客户端的测试



原文：http://rfyiamcool.blog.51cto.com/1030776/1380754

```
    //xiaorui.cc
package main
import(
"fmt"
"net/http"
"io/ioutil"
"log"
"time"
"redis"
)
//xiaorui.cc
const AddForm = `
<html><body>
<form method="POST" action="/add">
Name: <input type="text" name="name">
Age: <input type="text" name="age">
<input type="submit" value="Add">
</form>
</body></html>
`
const setform = `
<html><body>
<form method="POST" action="/set">
key: <input type="text" name="key">
value: <input type="text" name="value">
<input type="submit" value="set">
</form>
</body></html>
`
func Handler( w http.ResponseWriter,r *http.Request ){
path := r.URL.Path[1:]
if path == "favicon.ico"  {
```

```
        http.NotFound(w, r)
        return
}
if path == ""{
        path = "index.html"
}
contents,err:= ioutil.ReadFile( path )
if err !=nil{
        fmt.Fprintf( w,"404" )
        return
}
fmt.Fprintf( w,"%s\n",contents )
}

func Add( w http.ResponseWriter,r *http.Request ){
name := r.FormValue("name")
age := r.FormValue("age")
if name == "" || age == "" {
        fmt.Fprint(w, AddForm)
        return
}
fmt.Fprintf(w, "Save : Your name is   %s , You age is %s",name,age)
}
func redisset( w http.ResponseWriter,r *http.Request ){
key := r.FormValue("key")
value := r.FormValue("value")
if key == "" || value == "" {
        fmt.Fprint(w, setform)
        return
}
spec := redis.DefaultSpec().Db(0).Password("");
client, e := redis.NewSynchClientWithSpec (spec);
if e != nil { log.Println ("服务器连接有异常", e); return }
inva := []byte(value)
client.Set(key, inva);
fmt.Fprintf(w, "哥们，你输入的 key   %s 和 value   %s 已经插入到 redis 里面了",key,key)
}
func redisget( w http.ResponseWriter,r *http.Request ){
key := r.FormValue("key")
if key == "" {
        fmt.Fprint(w, setform)
        return
}
spec := redis.DefaultSpec().Db(0).Password("");
client, e := redis.NewSynchClientWithSpec (spec);
```

```
if e != nil { log.Println ("服务器连接有异常", e); return }

value, e := client.Get(key);
fmt.Fprintf(w, "哥们，你要查询的 key   %s 和 value   %s ",key,value)
}
func valueget(w http.ResponseWriter, r *http.Request) {
params := r.URL.Query()
user := params.Get("user")
fmt.Fprintf(w, "you are get user %s", user)
}

func main(){
http.HandleFunc( "/",Handler)
http.HandleFunc( "/add",Add)
http.HandleFunc( "/redisset",redisset)
http.HandleFunc( "/redisget",redisget)
http.HandleFunc( "/valueget",valueget)
s := &http.Server{
      Addr:                ":80",
      ReadTimeout:       30 * time.Second,
      WriteTimeout:     30 * time.Second,
      MaxHeaderBytes: 1 << 20,
}
log.Fatal(s.ListenAndServe())
}
```

Andriod 从源码的角度详解 View,ViewGroup 的 Touch 事件的分发机制

作者： xiaanming520 来源：http://xiaanming.blog.51cto.com/4991780/1382053

今天这篇文章主要分析的是 Android 的事件分发机制，采用例子加源码的方式让大家深刻的理解 Android 事件分发的具体情况，虽然网上很多 Android 的事件分发的文章，有些还写的不错，但是我还是决定写这篇文章，用我自己的思维方式来帮助大家理解 Android 事件分发，Android 事件分发到底有多重要呢？相信很多 Android 开发者都明白吧，这个我就不介绍了，我也写了很多篇文章里面涉及到 Android 的事件处理的问题，可能不理解 Android 事件分发的朋友会有点难理解吧，不过没关系，相信看了这篇文章的你会对 Android 事件分发有进一步的理解。我这篇文章分析的源码是 Android 2.2 的源码，也许你会说，干嘛不去分析最新的源码呢？我这里要解释一下，Android 2.2 的源码跟最新的源码在功能效果方面是一样的，只不过最新的源码相对于 Android 2.2 来说更加健壮一些， Android 2.2 的事件处理的代码几乎都写在一个方法体里面，而最新的源码分了很多个方法写，如果用最新的源码调用方法会绕来绕去的，相信你看的也会晕，出于这个考虑，我就拿 Android 2.2 的源码来给大家分析。

ViewGroup 的事件分发机制

我们用手指去触摸 Android 手机屏幕，就会产生一个触摸事件，但是这个触摸事件在底层是怎么分发的呢？这个我还真不知道，这里涉及到操作硬件（手机屏幕）方面的知识，也就是 Linux 内核方面的知识，我也没有了解过这方面的东西，所以我们可能就往上层来分析分析，我们知道 Android 中负责与用户交互，与用户操作紧密相关的四大组件之一是 Activity，所以我们有理由相信 Activity 中存在分发事件的方法，这个方法就是 dispatchTouchEvent(),我们先看其源码吧

```
public boolean dispatchTouchEvent(MotionEvent ev) {

    //如果是按下状态就调用 onUserInteraction()方法，onUserInteraction()方法
    //是个空的方法，我们直接跳过这里看下面的实现
    if (ev.getAction() == MotionEvent.ACTION_DOWN) {
        onUserInteraction();
    }

    if (getWindow().superDispatchTouchEvent(ev)) {
        return true;
    }

    //getWindow().superDispatchTouchEvent(ev)返回 false，这个事件就交给 Activity
    //来处理， Activity 的 onTouchEvent()方法直接返回了 false
    return onTouchEvent(ev);
}
```

这个方法中我们还是比较关心 getWindow()的 superDispatchTouchEvent()方法，getWindow()返回

当前 Activity 的顶层窗口 Window 对象，我们直接看 Window API 的 superDispatchTouchEvent()方法

```
/**
 * Used by custom windows, such as Dialog, to pass the touch screen event
 * further down the view hierarchy. Application developers should
 * not need to implement or call this.
 *
 */
public abstract boolean superDispatchTouchEvent(MotionEvent event);
```

这个是个抽象方法，所以我们直接找到其子类来看看 superDispatchTouchEvent()方法的具体逻辑实现，Window 的唯一子类是 PhoneWindow,我们就看看 PhoneWindow 的 superDispattEvent()

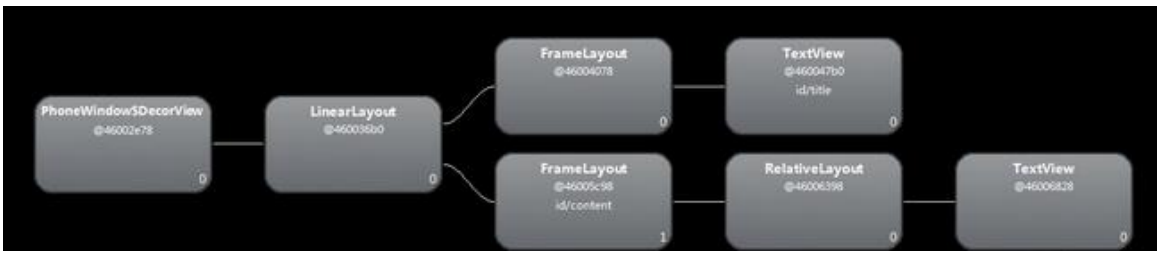
```
public boolean superDispatchTouchEvent(KeyEvent event) {
    return mDecor.superDispatchTouchEvent(event);
};
```

里面直接调用 DecorView 类的 superDispatchTouchEvent 方法，或许很多人不了解 DecorView 这个类，DecorView 是 PhoneWindow 的一个 final 的内部类并且继承 FrameLayout 的，也是 Window 界面的最顶层的 View 对象，这是什么意思呢？别着急，我们接着往下看

我们先新建一个项目，取名 AndroidTouchEvent，然后直接用模拟器运行项目， MainActivity 的布局文件为

```
    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity" >

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:text="@string/hello_world" />

</RelativeLayout>
```

利用 hierarchyviewer 工具来查看下 MainActivity 的 View 的层次结构，如下图



我们看到最顶层就是 PhoneWindow$DecorView，接着 DecorView 下面有一个 LinearLayout，LinearLayout 下面有两个 FrameLayout

上面那个 FrameLayout 是用来显示标题栏的，这个 Demo 中是一个 TextView,当然我们还可以定制我们的标题栏，利用 getWindow().setFeatureInt(Window.FEATURE_CUSTOM_TITLE,R.layout.XXX); xxx 就是我们自定义标题栏的布局 XML 文件

下面的 FrameLayout 是用来装载 ContentView 的，也就是我们在 Activity 中利用 setContentView()方法设置的 View，现在我们知道了，原来我们利用 setContentView()设置 Activity 的 View 的外面还嵌套了这么多的东西

我们来理清下思路，Activity 的最顶层窗体是 PhoneWindow,而 PhoneWindow 的最顶层 View 是 DecorView，接下来我们就看 DecorView 类的 superDispatchTouchEvent()方法

```
public boolean superDispatchTouchEvent(MotionEvent event) {
        return super.dispatchTouchEvent(event);
    }
```

在里面调用了父类 FrameLayout 的 dispatchTouchEvent()方法，而 FrameLayout 中并没有 dispatchTouchEvent()方法，所以我们直接看 ViewGroup 的 dispatchTouchEvent()方法

```
/**
* {@inheritDoc}
    */
   @Override
   public boolean dispatchTouchEvent(MotionEvent ev) {
    final int action = ev.getAction();
    final float xf = ev.getX();
    final float yf = ev.getY();
    final float scrolledXFloat = xf + mScrollX;
    final float scrolledYFloat = yf + mScrollY;
    final Rect frame = mTempRect;

    //这个值默认是 false，然后我们可以通过 requestDisallowInterceptTouchEvent(boolean disallowIntercept)方法
    //来改变 disallowIntercept 的值
    boolean disallowIntercept = (mGroupFlags & FLAG_DISALLOW_INTERCEPT) != 0;
```

```
//这里是 ACTION_DOWN 的处理逻辑
if (action == MotionEvent.ACTION_DOWN) {
 //清除 mMotionTarget，每次 ACTION_DOWN 都很设置 mMotionTarget 为 null
    if (mMotionTarget != null) {
        mMotionTarget = null;
    }

    //disallowIntercept 默认是 false，就看 ViewGroup 的 onInterceptTouchEvent()方法
    if (disallowIntercept || !onInterceptTouchEvent(ev)) {
        ev.setAction(MotionEvent.ACTION_DOWN);
        final int scrolledXInt = (int) scrolledXFloat;
        final int scrolledYInt = (int) scrolledYFloat;
        final View[] children = mChildren;
        final int count = mChildrenCount;
        //遍历其子 View
        for (int i = count - 1; i >= 0; i--) {
            final View child = children[i];

            //如果该子 View 是 VISIBLE 或者该子 View 正在执行动画，表示该 View 才
            //可以接受到 Touch 事件
            if ((child.mViewFlags & VISIBILITY_MASK) == VISIBLE
                    || child.getAnimation() != null) {
             //获取子 View 的位置范围
                child.getHitRect(frame);

                //如 Touch 到屏幕上的点在该子 View 上面
                if (frame.contains(scrolledXInt, scrolledYInt)) {
                    // offset the event to the view's coordinate system
                    final float xc = scrolledXFloat - child.mLeft;
                    final float yc = scrolledYFloat - child.mTop;
                    ev.setLocation(xc, yc);
                    child.mPrivateFlags &= ~CANCEL_NEXT_UP_EVENT;

                    //调用该子 View 的 dispatchTouchEvent()方法
                    if (child.dispatchTouchEvent(ev))  {
                        //  如果 child.dispatchTouchEvent(ev)返回 true 表示
                     //该事件被消费了，设置 mMotionTarget 为该子 View
                        mMotionTarget = child;
                        //直接返回 true
                        return true;
                    }
                    // The event didn't get handled, try the next view.
                    // Don't reset the event's location, it's not
                    // necessary here.
                }
```

```
                }
            }
        }
    }

    //判断是否为 ACTION_UP 或者 ACTION_CANCEL
    boolean isUpOrCancel = (action == MotionEvent.ACTION_UP) ||
            (action == MotionEvent.ACTION_CANCEL);

    if (isUpOrCancel) {
        //如果是 ACTION_UP 或者 ACTION_CANCEL, 将 disallowIntercept 设置为默认的 false
     //假如我们调用了 requestDisallowInterceptTouchEvent()方法来设置 disallowIntercept 为 true
     //当我们抬起手指或者取消 Touch 事件的时候要将 disallowIntercept 重置为 false
     //所以说上面的 disallowIntercept 默认在我们每次 ACTION_DOWN 的时候都是 false
        mGroupFlags &= ~FLAG_DISALLOW_INTERCEPT;
    }

    // The event wasn't an ACTION_DOWN, dispatch it to our target if
    // we have one.
    final View target = mMotionTarget;
    //mMotionTarget 为 null 意味着没有找到消费 Touch 事件的 View, 所以我们需要调用 ViewGroup 父类的
    //dispatchTouchEvent()方法，也就是 View 的 dispatchTouchEvent()方法
    if (target == null) {
        // We don't have a target, this means we're handling the
        // event as a regular view.
        ev.setLocation(xf, yf);
        if ((mPrivateFlags & CANCEL_NEXT_UP_EVENT) != 0) {
            ev.setAction(MotionEvent.ACTION_CANCEL);
            mPrivateFlags &= ~CANCEL_NEXT_UP_EVENT;
        }
        return super.dispatchTouchEvent(ev);
    }

    //这个 if 里面的代码 ACTION_DOWN 不会执行，只有 ACTION_MOVE
    //ACTION_UP 才会走到这里, 假如在 ACTION_MOVE 或者 ACTION_UP 拦截的
    //Touch 事件, 将 ACTION_CANCEL 派发给 target，然后直接返回 true
    //表示消费了此 Touch 事件
    if (!disallowIntercept && onInterceptTouchEvent(ev)) {
        final float xc = scrolledXFloat - (float) target.mLeft;
        final float yc = scrolledYFloat - (float) target.mTop;
        mPrivateFlags &= ~CANCEL_NEXT_UP_EVENT;
        ev.setAction(MotionEvent.ACTION_CANCEL);
        ev.setLocation(xc, yc);

        if (!target.dispatchTouchEvent(ev)) {
```

```
            }
            // clear the target
            mMotionTarget = null;
            // Don't dispatch this event to our own view, because we already
            // saw it when intercepting; we just want to give the following
            // event to the normal onTouchEvent().
            return true;
        }

        if (isUpOrCancel) {
            mMotionTarget = null;
        }

        // finally offset the event to the target's coordinate system and
        // dispatch the event.
        final float xc = scrolledXFloat - (float) target.mLeft;
        final float yc = scrolledYFloat - (float) target.mTop;
        ev.setLocation(xc, yc);

        if ((target.mPrivateFlags & CANCEL_NEXT_UP_EVENT) != 0) {
            ev.setAction(MotionEvent.ACTION_CANCEL);
            target.mPrivateFlags &= ~CANCEL_NEXT_UP_EVENT;
            mMotionTarget = null;
        }

        //如果没有拦截 ACTION_MOVE, ACTION_DOWN 的话，直接将 Touch 事件派发给 target
        return target.dispatchTouchEvent(ev);
    }
```

这个方法相对来说还是蛮长，不过所有的逻辑都写在一起，看起来比较方便，接下来我们就具体来分析一下

我们点击屏幕上面的 TextView 来看看 Touch 是如何分发的，先看看 ACTION_DOWN

在 DecorView 这一层会直接调用 ViewGroup 的 dispatchTouchEvent(), 先看 18 行，每次 ACTION_DOWN 都会将 mMotionTarget 设置为 null, mMotionTarget 是什么？我们先不管，继续看代码，走到 25 行， disallowIntercept 默认为 false，我们再看 ViewGroup 的 onInterceptTouchEvent()方法

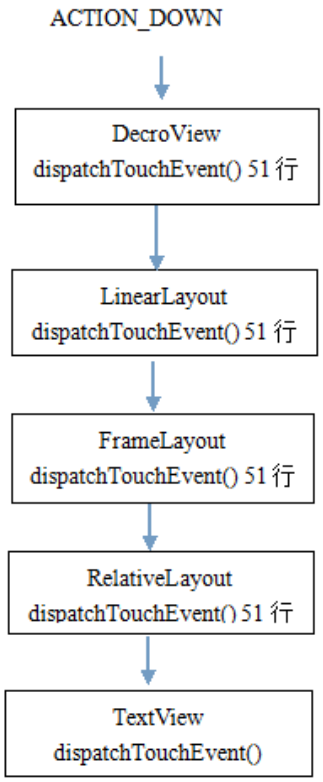
```
public boolean onInterceptTouchEvent(MotionEvent ev) {
    return false;
}
```

直接返回 false，继续往下看，循环遍历 DecorView 里面的 Child，从上面的 MainActivity 的层次结构图我们可以看出，DecorView 里面只有一个 Child 那就是 LinearLayout，第 43 行判断 Touch 的位置在不在 LinnearLayout 上面，这是毫无疑问的，所以直接跳到 51 行， 调用 LinearLayout 的 dispatchTouchEvent()方法，LinearLayout 也没有 dispatchTouchEvent()这个方法，所以也是调用 ViewGroup 的 dispatchTouchEvent()方法，所以这个方法卡在 51 行没有继续下去，而是去先执行 LinearLayout 的 dispatchTouchEvent()

LinearLayout 调用 dispatchTouchEvent()的逻辑跟 DecorView 是一样的 ,所以也是遍历 LinearLayout 的两个 FrameLayout，判断 Touch 的是哪个 FrameLayout，很明显是下面那个，调用下面那个 FrameLayout 的 dispatchTouchEvent()， 所以 LinearLayout 的 dispatchTouchEvent()卡在 51 也没继续下去

继续调用 FrameLayout 的 dispatchTouchEvent()方法，和上面一样的逻辑，下面的 FrameLayout 也只有一个 Child，就是 RelativeLayout，FrameLayout 的 dispatchTouchEvent()继续卡在 51 行，先执行 RelativeLayout 的 dispatchTouchEvent()方法

执行 RelativeLayout 的 dispatchTouchEvent()方法逻辑还是一样的，循环遍历 RelativeLayout 里面的孩子，里面只有一个 TextView，所以这里就调用 TextView 的 dispatchTouchEvent()， TextView 并没有 dispatchTouchEvent()这个方法 ,于是找 TextView 的父类 View ,在看 View 的 dispatchTouchEvent()的方法之前，我们先理清下上面这些 ViewGroup 执行 dispatchTouchEvent()的思路，我画了一张图帮大家理清下（这里没有画出 onInterceptTouchEvent（ ）方法）



上面的 ViewGroup 的 Touch 事件分发就告一段落先，因为这里要调用 TextView（也就是 View）的 dispatchTouchEvent()方法，所以我们先分析 View 的 dispatchTouchEvent()方法在将上面的继续下去

View 的 Touch 事件分发机制

我们还是先看 View 的 dispatchTouchEvent()方法的源码

```
public boolean dispatchTouchEvent(MotionEvent event) {
    if (mOnTouchListener != null && (mViewFlags & ENABLED_MASK) == ENABLED &&
            mOnTouchListener.onTouch(this, event)) {
        return true;
    }
    return onTouchEvent(event);
}
```

在这个方法里面，先进行了一个判断

第一个条件 mOnTouchListener 就是我们调用 View 的 setTouchListener()方法设置的
第二个条件是判断 View 是否为 enabled 的， View 一般都是 enabled,除非你手动设置为 disabled
第三个条件就是 OnTouchListener 接口的 onTouch()方法的返回值了，如果调用了 setTouchListener()设置 OnTouchListener，并且 onTouch()方法返回 true，View 的 dispatchTouchEvent()方法就直接返回 true,否则就执行 View 的 onTouchEvent() 并返回 View 的 onTouchEvent()的值

现在你了解了 View 的 onTouchEvent()方法和 onTouch()的关系了吧,为什么 Android 提供了处理 Touch 事件 onTouchEvent()方法还要增加一个 OnTouchListener 接口呢？我觉得 OnTouchListener 接口是对处理 Touch 事件的屏蔽和扩展作用吧，屏蔽作用我就不举例介绍了，看上面的源码就知道了，我就说下扩展吧，比如我们要打印 View 的 Touch 的点的坐标，我们可以自定义一个 View 如下：

```
public class CustomView extends View {

public CustomView(Context context, AttributeSet attrs) {
    super(context, attrs);
}

public CustomView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
}

@Override
public boolean onTouchEvent(MotionEvent event) {

    Log.i("tag", "X 的坐标 = " + event.getX() + " Y 的坐标 = " + event.getY());

    return super.onTouchEvent(event);
```

```
}

}
```

也可以直接对 View 设置 OnTouchListener 接口,在 return 的时候调用下 v.onTouchEvent()

```
view.setOnTouchListener(new OnTouchListener() {

        @Override
        public boolean onTouch(View v, MotionEvent event) {

            Log.i("tag", "X 的坐标 = " + event.getX() + " Y 的坐标 = " + event.getY());

            return v.onTouchEvent(event);
        }
    });
```

这样子也实现了我们所需要的功能，所以我认为 OnTouchListener 是对 onTouchEvent()方法的一个屏蔽和扩展作用，假如你有不一样的理解，你也可以告诉我下，这里就不纠结这个了。

我们再看 View 的 onTouchEvent()方法

```
public boolean onTouchEvent(MotionEvent event) {
  final int viewFlags = mViewFlags;

  if ((viewFlags & ENABLED_MASK) == DISABLED) {
      return (((viewFlags & CLICKABLE) == CLICKABLE ||
              (viewFlags & LONG_CLICKABLE) == LONG_CLICKABLE));
  }

  //如果设置了 Touch 代理，就交给代理来处理，mTouchDelegate 默认是 null
  if (mTouchDelegate != null) {
      if (mTouchDelegate.onTouchEvent(event)) {
          return true;
      }
  }

  //如果 View 是 clickable 或者 longClickable 的 onTouchEvent 就返回 true，否则返回 false
  if (((viewFlags & CLICKABLE) == CLICKABLE ||
          (viewFlags & LONG_CLICKABLE) == LONG_CLICKABLE)) {
      switch (event.getAction()) {
          case MotionEvent.ACTION_UP:
              boolean prepressed = (mPrivateFlags & PREPRESSED) != 0;
              if ((mPrivateFlags & PRESSED) != 0 || prepressed) {
```

```
                boolean focusTaken = false;
                if (isFocusable() && isFocusableInTouchMode() && !isFocused()) {
                    focusTaken = requestFocus();
                }

                if (!mHasPerformedLongPress) {
                    removeLongPressCallback();

                    if (!focusTaken) {
                        if (mPerformClick == null) {
                            mPerformClick = new PerformClick();
                        }
                        if (!post(mPerformClick)) {
                            performClick();
                        }
                    }
                }

                if (mUnsetPressedState == null) {
                    mUnsetPressedState = new UnsetPressedState();
                }

                if (prepressed) {
                    mPrivateFlags |= PRESSED;
                    refreshDrawableState();
                    postDelayed(mUnsetPressedState,
                            ViewConfiguration.getPressedStateDuration());
                } else if (!post(mUnsetPressedState)) {
                    mUnsetPressedState.run();
                }
                removeTapCallback();
            }
            break;

        case MotionEvent.ACTION_DOWN:
            if (mPendingCheckForTap == null) {
                mPendingCheckForTap = new CheckForTap();
            }
            mPrivateFlags |= PREPRESSED;
            mHasPerformedLongPress = false;
            postDelayed(mPendingCheckForTap, ViewConfiguration.getTapTimeout());
            break;

        case MotionEvent.ACTION_CANCEL:
            mPrivateFlags &= ~PRESSED;
```

```
                refreshDrawableState();
                removeTapCallback();
                break;

            case MotionEvent.ACTION_MOVE:
                final int x = (int) event.getX();
                final int y = (int) event.getY();

                //当手指在 View 上面滑动超过 View 的边界，
                int slop = mTouchSlop;
                if ((x < 0 - slop) || (x >= getWidth() + slop) ||
                        (y < 0 - slop) || (y >= getHeight() + slop)) {
                    // Outside button
                    removeTapCallback();
                    if ((mPrivateFlags & PRESSED) != 0) {
                        removeLongPressCallback();

                        mPrivateFlags &= ~PRESSED;
                        refreshDrawableState();
                    }
                }
                break;
        }
        return true;
    }

    return false;
```

这个方法也是比较长的，我们先看第 4 行，如果一个 View 是 disabled，并且该 View 是 Clickable 或者 longClickable， onTouchEvent()就不执行下面的代码逻辑直接返回 true，表示该 View 就一直消费 Touch 事件，如果一个 enabled 的 View,并且是 clickable 或者 longClickable 的，onTouchEvent()会执行下面的代码逻辑并返回 true，综上，一个 clickable 或者 longclickable 的 View 是一直消费 Touch 事件的，而一般的 View 既不是 clickable 也不是 longclickable 的(即不会消费 Touch 事件，只会执行 ACTION_DOWN 而不会执行 ACTION_MOVE 和 ACTION_UP) Button 是 clickable 的，可以消费 Touch 事件，但是我们可以通过 setClickable()和 setLongClickable()来设置 View 是否为 clickable 和 longClickable。当然还可以通过重写 View 的 onTouchEvent()方法来控制 Touch 事件的消费与否

我们在看 57 行的 ACTION_DOWN，新建一个 CheckForTap，我们看看 CheckForTap 是什么

```
private final class CheckForTap implements Runnable {
    public void run() {
        mPrivateFlags &= ~PREPRESSED;
        mPrivateFlags |= PRESSED;
        refreshDrawableState();
```

```
            if ((mViewFlags & LONG_CLICKABLE) == LONG_CLICKABLE) {
                postCheckForLongClick(ViewConfiguration.getTapTimeout());
            }
        }
    }
```

原来是个 Runnable 对象 然后使用 Handler 的 post 方法延时 ViewConfiguration.getTapTimeout()执行 CheckForTap 的 run()方法 ,在 run 方法中先判断 view 是否 longClickable 的,一般的 View 都是 false, postCheckForLongClick(ViewConfiguration.getTapTimeout())这段代码就是执行长按的逻辑的代码，只有当我们设置为 longClickble 才会去执行 postCheckForLongClick(ViewConfiguration.getTapTimeout())，这里我就不介绍了

由于考虑到文章篇幅的问题，我就不继续分析 View 的长按事件和点击事件了，在这里我直接得出结论吧

长按事件是在 ACTION_DOWN 中执行，点击事件是在 ACTION_UP 中执行，要想执行长按事件，这个 View 必须是 longclickable 的， 也许你会纳闷，一般的 View 不是 longClickable 为什么也会执行长按事件呢？我们要执行长按事件必须要调用 setOnLongClickListener()设置 OnLongClickListener 接口，我们看看这个方法的源码

```
public void setOnLongClickListener(OnLongClickListener l) {
 if (!isLongClickable()) {
      setLongClickable(true);
 }
 mOnLongClickListener = l;
}l 看到没有，如果这个 View 不是 longClickable 的，我们就调用 setLongClickable(true)方法设置为 ongClickable 的,所以才会去执行长按方法 onLongClick();
```

要想执行点击事件，这个 View 就必须要消费 ACTION_DOWN 和 ACTION_MOVE 事件，并且没有设置 OnLongClickListener 的情况下，如果设置了 OnLongClickListener 的情况下，需要 onLongClick()返回 false 才能执行到 onClick()方法，也许你又会纳闷，一般的 View 默认是不消费 touch 事件的，这不是和你上面说的相违背嘛，我们要向执行点击事件必须要调用 setOnClickListener()来设置 OnClickListener 接口，我们看看这个方法的源码就知道了

```
public void setOnClickListener(OnClickListener l) {
 if (!isClickable()) {
      setClickable(true);
 }
 mOnClickListener = l;
```

```
}
```

所以说一个 enable 的并且是 clickable 的 View 是一直消费 touch 事件的，所以才会执行到 onClick（）方法

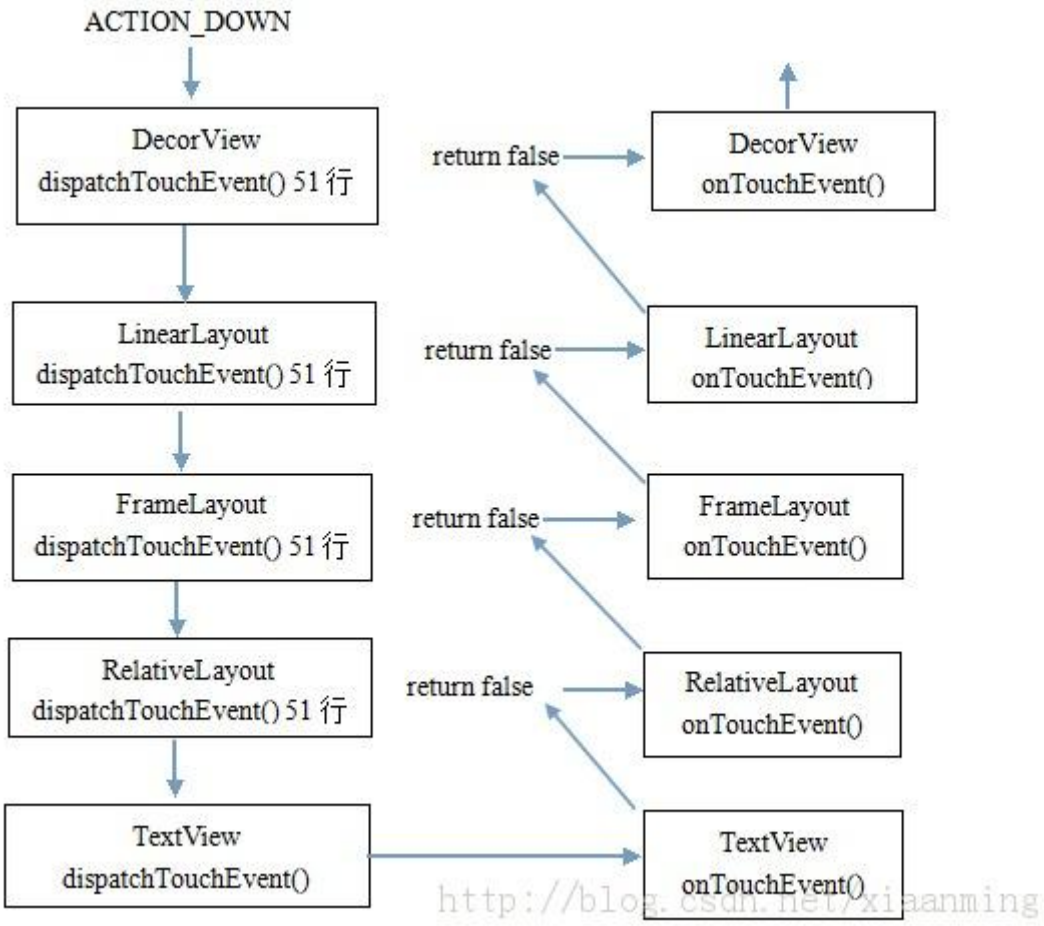
对于 View 的 Touch 事件的分发机制算是告一段落了，从上面我们可以得出 TextView 的

dispatchTouchEvent()的返回 false 的,即不消费 Touch 事件。我们就要往上看 RelativeLayout 的 dispatchTouchEvent()方法的 51 行，由于 TextView.dispatchTouchEvent()为 false，导致 mMotionTarget 没有被赋值 ,还是 null，继续往下走执行 RelativeLayout 的 dispatchTouchEvent()方法，来到第 84 行， 判断 target 是否为 null，这个 target 就是 mMotionTarget，满足条件，执行 92 行的 super.dispatchTouchEvent(ev)代码并返回， 这里调用的是 RelativeLayout 父类 View 的 dispatchTouchEvent()方法，由于 RelativeLayout 没有设置 onTouchListener，所以这里直接调用

RelativeLayout(其实就是 View，因为 RelativeLayout 没有重写 onTouchEvent())的 onTouchEvent()方法 由于 RelativeLayout 既不是 clickable 的也是 longClickable 的 ,所以其 onTouchEvent()方法 false，RelativeLayout 的 dispatchTouchEvent()也是返回 false,这里就执行完了 RelativeLayout 的 dispatchTouchEvent()方法

继续执行 FrameLayout 的 dispatchTouchEvent()的第 51 行 ,由于 RelativeLayout.dispatchTouchEvent()返回的是 false，跟上面的逻辑是一样的，也是执行到 92 行的 super.dispatchTouchEvent(ev)代码并返回，然后执行 FrameLayout 的 onTouchEvent()方法，而 FrameLayout 的 onTouchEvent()也是返回 false,所以 FrameLayout 的 dispatchTouchEvent()方法返回 false,执行完毕 FrameLayout 的 dispatchTouchEvent()方法

在上面的我就不分析了，大家自行分析一下，跟上面的逻辑是一样的，我直接画了个图来帮大家理解下（这里没有画出 onInterceptTouchEvent（）方法）

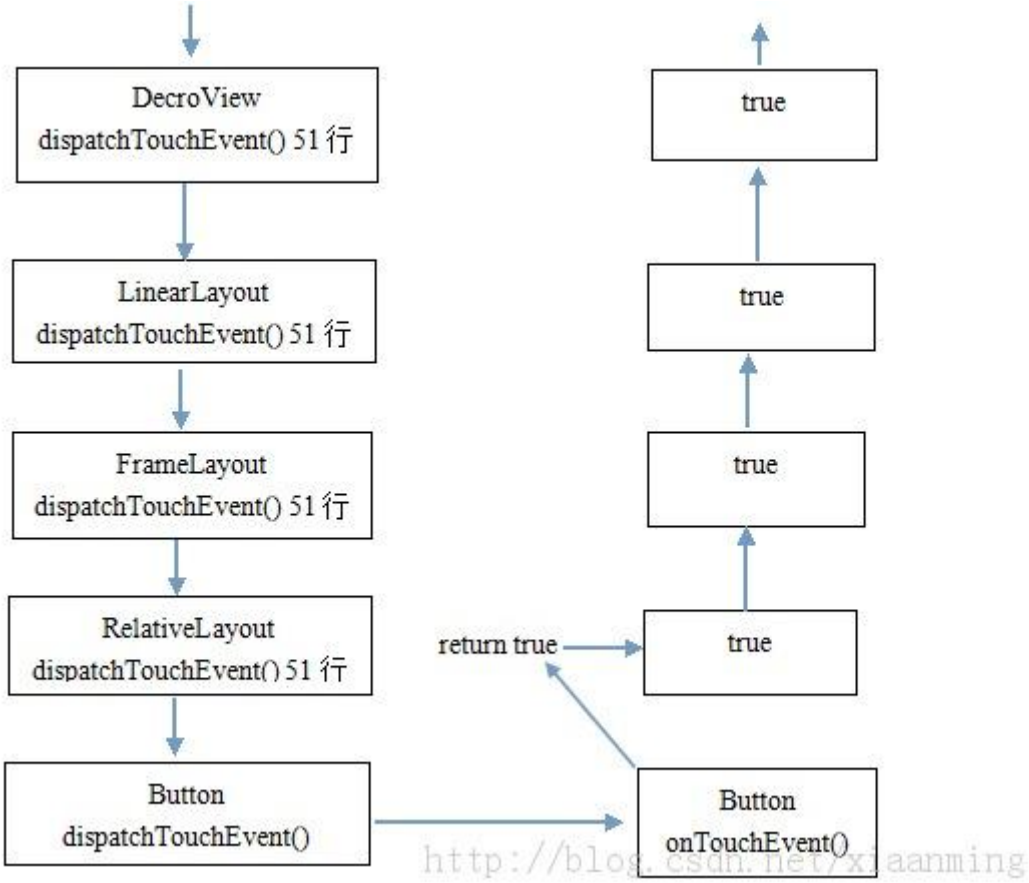


所以我们点击屏幕上面的 TextView 的事件分发流程是上图那个样子的，表示 Activity 的 View 都不消费 ACTION_DOWN 事件，所以就不能在触发 ACTION_MOVE, ACTION_UP 等事件了，具体是为什么？我还不太清楚，毕竟从 Activity 到 TextView 这一层是分析不出来的，估计是在底层实现的。

但如果将 TextView 换成 Button，流程是不是还是这个样子呢？答案不是,我们来分析分析一下，如果是 Button，Button 是一个 clickable 的 View，onTouchEvent()返回 true，表示他一直消费 Touch 事件，所以 Button 的 dispatchTouchEvent()方法返回 true，回到 RelativeLayout 的 dispatchTouchEvent()方法的 51 行，满足条件，进入到 if 方法体，设置 mMotionTarget 为 Button，然后直接返回 true，RelativeLayout 的 dispatchTouchEvent()方法执行完毕，不会调用到 RelativeLayout 的 onTouchEvent()方法

然后到 FrameLayout 的 dispatchTouchEvent()方法的 51 行，由于 RelativeLayout.dispatchTouchEvent()返回 true，满足条件，进入 if 方法体，设置 mMotionTarget 为 RelativeLayout，注意下，这里的 mMotionTarget 跟 RelativeLayout 的 dispatchTouchEvent()方法的 mMotionTarget 不是同一个哦，因为他们是不同的方法中的，然后返回 true

同理 FrameLayout 的 dispatchTouchEvent()也是返回 true，DecorView 的 dispatchTouchEvent()方法也返回 true， 还是画一个流程图（这里没有画出 onInterceptTouchEvent（ ）方法）给大家理清下



从上面的流程图得出一个结论，Touch 事件是从顶层的 View 一直往下分发到手指按下的最里面的 View，如果这个 View 的 onTouchEvent()返回 false,即不消费 Touch 事件，这个 Touch 事件就会向上找父布局调用其父布局的 onTouchEvent()处理，如果这个 View 返回 true,表示消费了 Touch 事件，就不调用父布局的 onTouchEvent()

接下来我们用一个自定义的 ViewGroup 来替换 RelativeLayout,自定义 ViewGroup 代码如下

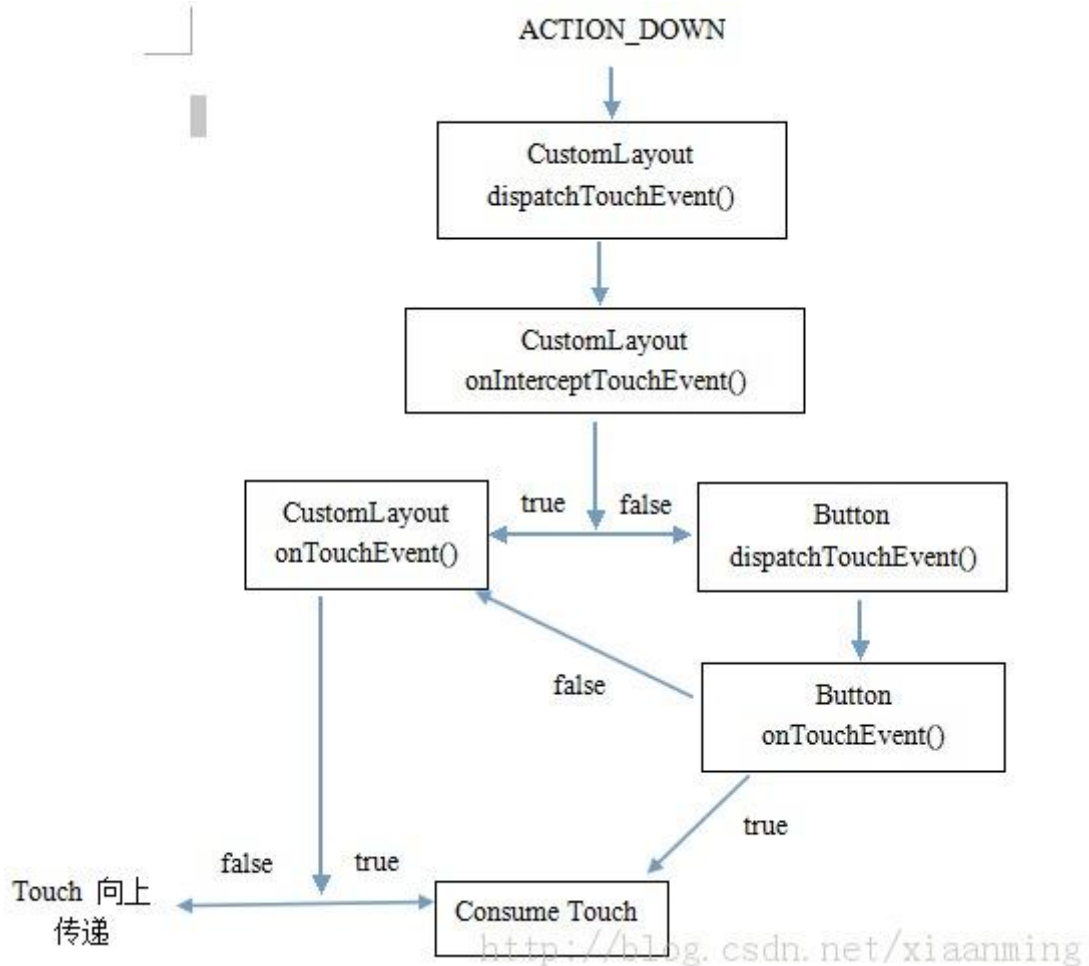
```
package com.example.androidtouchevent;

import android.content.Context;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.widget.RelativeLayout;

public class CustomLayout extends RelativeLayout {

public CustomLayout(Context context, AttributeSet attrs) {
    super(context, attrs, 0);
}

public CustomLayout(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
}
```

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    return super.onTouchEvent(event);
}

@Override
public boolean onInterceptTouchEvent(MotionEvent ev) {
    return true;
}


}
```

我们就重写了 onInterceptTouchEvent(),返回 true, RelativeLayout 默认是返回 false, 然后再 CustomLayout 布局中加一个 Button ,如下图

我们这次不从 DecorView 的 dispatchTouchEvent()分析了，直接从 CustomLayout 的 dispatchTouchEvent()分析

我们先看 ACTION_DOWN 来到 25 行，由于我们重写了 onInterceptTouchEvent()返回 true， 所以不走这个 if 里面，直接往下看代码,来到 84 行， target 为 null,所以进入 if 方法里面，直接调用 super.dispatchTouchEvent()方法， 也就是 View 的 dispatchTouchEvent()方法，而在 View 的 dispatchTouchEvent()方法中是直接调用 View 的 onTouchEvent()方法，但是 CustomLayout 重写了 onTouchEvent()，所以这里还是调用 CustomLayout 的 onTouchEvent()， 这个方法返回 false，不消费 Touch 事件，所以不会在触发 ACTION_MOVE,ACTION_UP 等事件了，这里我再画一个流程图吧（含有 onInterceptTouchEvent()方法的）



好了，就分析到这里吧，差不多分析完了，还有一种情况没有分析到，例如我将 CustomLayout 的代码改成下面的情形，Touch 事件又是怎么分发的呢？我这里就不带大家分析了

```
package com.example.androidtouchevent;

import android.content.Context;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.widget.RelativeLayout;

public class CustomLayout extends RelativeLayout {
```

```
public CustomLayout(Context context, AttributeSet attrs) {
    super(context, attrs, 0);
}

public CustomLayout(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
}

@Override
public boolean onTouchEvent(MotionEvent event) {
    return super.onTouchEvent(event);
}

@Override
public boolean onInterceptTouchEvent(MotionEvent ev) {
    if(ev.getAction() == MotionEvent.ACTION_MOVE){
        return true;
    }
    return super.onInterceptTouchEvent(ev);
}


}
```

这篇文章的篇幅有点长，如果你想了解 Touch 事件的分发机制，你一定要认真看完，下面来总结一下吧

1.Activity 的最顶层 Window 是 PhoneWindow，PhoneWindow 的最顶层 View 是 DecorView

2.一个 clickable 或者 longClickable 的 View 会永远消费 Touch 事件，不管他是 enabled 还是 disabled 的

3.View 的长按事件是在 ACTION_DOWN 中执行，要想执行长按事件该 View 必须是 longClickable 的，并且不能产生 ACTION_MOVE

4.View 的点击事件是在 ACTION_UP 中执行，想要执行点击事件的前提是消费了 ACTION_DOWN 和 ACTION_MOVE,并且没有设置 OnLongClickListener 的情况下 如设置了 OnLongClickListener 的情况，则必须使 onLongClick()返回 false

5.如果 View 设置了 onTouchListener 了，并且 onTouch()方法返回 true，则不执行 View 的 onTouchEvent()方法，也表示 View 消费了 Touch 事件，返回 false 则继续执行 onTouchEvent()

6.Touch 事件是从最顶层的 View 一直分发到手指 touch 的最里层的 View,如果最里层 View 消费了 ACTION_DOWN 事件（设置 onTouchListener，并且 onTouch()返回 true 或者 onTouchEvent()方法返回 true）才会触发 ACTION_MOVE,ACTION_UP 的发生,如果某个 ViewGroup 拦截了 Touch 事件，则 Touch 事件交给 ViewGroup 处理

7.Touch 事件的分发过程中，如果消费了 ACTION_DOWN,而在分发 ACTION_MOVE 的时候,某个 ViewGroup 拦截了 Touch 事件，就像上面那个自定义 CustomLayout，则会将 ACTION_CANCEL 分发给该 ViewGroup 下面的 Touch 到的 View,然后将 Touch 事件交给 ViewGroup 处理，并返回 tru

在成功道路上，你要百败百战

作者：孙继滨 来源：http://sunjibin.blog.51cto.com/918334/1371462

前两年，我碰到一位十多年未见的高中同学。这位同学名校毕业后，一直脚踏实地，眼下已经是名企高管，事业蒸蒸日上。在星巴克，我谈起自己过往的疲惫孤独和颠沛流离。言语间，对他的坚实稳健以及成功事业，那真是化不开的嫉妒啊。结果，同学大怒而去。原来我浪迹天涯的生活，正是他一直想做而不敢做的。我对他的羡慕，反倒被他理解为讽刺，我对自己的不满，反倒被他理解为孔雀炫耀自己的尾巴。围城，无处不在啊。

失败，并不重要。重要的是过程。在你为失败而自怨自艾时，许多人羡慕你的自由的抉择和肆意的挥洒。失败，还是“失败”，真的很难说清。

关于失败，上面的说法已经是老生常谈了。下面，我要讲讲另一种失败。一种无法用上面说法粉饰的失败。

前两年读过一本书，名叫《不要一个人吃饭》。算是一个美国小子的成功学和自我推广。其中有一个细节令我印象深刻。此人以人际关系为核心竞争力，然而，当他在校园初步成功后，竟然自高自大目中无人，得罪了身边几乎所有人。可以说，他彻底失败了。这个时候，有两种选择，一个是放弃以人际关系为核心，改弦更张重新做人；另一个是走出现有环境、离开现有圈子，重新来过。他选择了后者，重建了自己的人际关系网。当然，最后他成了真正的人际关系达人。

读中国历史，不知道大家有没有感触。中国人不是屡教不改、拒绝接受新鲜事物。而是教训记得太牢，矫枉过正。唐末五代十国，军阀乱战，武人为非作歹。宋代就改成文人治国，重文抑武，一改几百年，以至于此后中国武功一直不盛，两次为外族灭国。这就是矫枉过正了。上世纪五十年代，出于某些原因和问题，政府取缔了证券银行保险行业。那些所谓原因和问题，现在也是存在，但至于取缔这些行业吗？

每个人都有自己的能力，围绕这个能力，人们不断学习知识、积累资源，获取经验。在工作中，从投入精力到获得收入，我们总要经过一个路径。这个路径，一般的说法是赚钱路径；如果自己的最大能力也在路径上，这个路径就叫做成功路径。比方说，史玉柱的最大能力就是营销能力。只要是能发挥营销能力的工作，无论是哪个行业，都在史玉柱的成功路径上。史玉柱也会编程，他也可以用编程赚钱，这是他的赚钱路径。

没有人不会失败。但是，成功路径上的失败，和赚钱路径上的失败，应该如何对待？是否需要区别对待？如果史玉柱在巨人大厦失败后，穷的只剩下几百块钱，最后老老实实地去当编程人员，踏踏实实地生活，是不是就对呢？这真是一个值得我们思考的问题。

有大学生朋友问我：我们看过许多励志书，激励自己，寻找成功的方法，书看完了，道理也懂了，但行动却跟不上，您觉得道理人人都懂，但成功和平凡到底差在哪里？我的回答是：成功就是发现、使用、磨练、提高自己的能力，赚钱是副产品，也就是走成功路径；平凡就是赚钱第一，什么赚钱做什么、什么赚钱快做什么，不能全身心投入在某项能力上，走的是赚钱路径。

有时生活对你的沉重打击让你措手不及，不要丧失信心。如今我确信，一个人之所以能够一直前进，

不是由于他喜欢自己所做的事情，而是由于他处在自己的成功路径上。所以，你要去发现自己的成功路径。如果失败发生，那很好。它给你一个机会去审视：我是在赚钱路径上，还是在成功路径上？如果是前者，你不妨将失败当做是美好的回忆，认输出局；如果是后者，你必须鼓起全身心的力量，狠狠地踩过去。

有些失败是不能认输的，可以百战百败，但是，必须百败百战。

手游公司运维之从一个人运维到运维主管

作者：自由 linux 来源：http://john88wang.blog.51cto.com/2165294/1370276

去年阴差阳错地进入了我目前的公司，一家定位于创业型公司的手游公司，公司目前有一款游戏在线运营，还有两款游戏在内测和开发当中。回首在这家公司工作的这段时光，感慨颇深，半年时间掌握的知识是我之前两年多掌握的还要多，在这边工作半年好像是工作了一年半。以前我在大公司的时候，觉得工作时间太清闲，想要获得更大的发展空间，想要学到更多的时候，更重要的是在大公司，没有存在感和成就感。在大公司，很多东西做得很完善了，我只需要去适应大公司里面的工作流程，去做那些已经做得很模块化的工作，但是他们是怎么样从一团混乱发展成流程标准化，我不得而知，也没人讲解过，当时就觉得在这家大公司里面，很多东西都不是我做的，我只是一名简单的运维小工，做些常规的简单操作，于是，我用尽心思去了解运维方面的底层架构，我当时在想，如果我有机会，我一定会亲手把一个公司的运维部发展成这家大公司的模式。

我去目前公司面试的当天收到两个 offer，一个是另外一家游戏公司，有一定规模，公司大概几层楼，有几百人。还有就是当前这家公司。这家公司的情况是刚成立不久，收购了另外一家公司的项目组，收购的项目刚开始运营，公司缺一个运维人员，但是干得工作比较杂乱，统管公司的所有的运维工作，又当网管又要管理线上的服务器。当二老板面试我的时候，说我之前一直做的服务器运维工作，但是在这边既要当网管又要做游戏运维时，我想了想，还是很坚定地说愿意到这边来工作。当时我想，虽然搞桌面维护是我非常不愿意干的事情，但是公司正处于发展阶段，刚开始难免会苦一点，只要过了这个阶段，我的发展空间将会很大。一周过后，入职的第一天就和二老板去搬电脑，入职第二天修电脑和安装操作系统，第三天就要直接管理线上的服务器。总之，之后的每天都是忙忙碌碌，好多天都是早上上班带的早餐，等到 10 点过才顾得上吃，不是帮同事处理桌面服务就是和开发讨论游戏上线的问题，由于一些原因，项目组原来的成员陆续都离职，包括策划，美术，客户端和服务端，很多部门新来的同事都是不到一周就要被逼接手工作，公司一团混乱，新来的策划还没有熟悉游戏资源配置，新来的服务端程序还没有熟悉游戏代码，运营那边就催促要更新资源，要发布上线。很长一段时间大家都是处于赶鸭子上架的状态，硬着头皮上，慢慢去摸索。除了工作交接的问题外，更可气的是原项目组成员隔三差五就捣乱，导致我们经常加班，四处救火。最严重的一次就是去年国庆之前的一天，公司自主运营的几个游戏区服玩家突然都进入不了游戏。这可急坏了我们，以为游戏服务器遭攻击了，或者有竞争对手捣乱，以为是 Nginx，PHP-FPM 参数没有配置适当，以为是 MongoDB 数据库不稳定，又请 ucloud 的技术支持帮忙处理，到当天晚上 1 点过，最终确定为 MongoDB 索引丢失导致 PHP 代码查询 MongoDB 数据库连接超时，进而玩家无法进入游戏。我和几个程序员都快要哭了的感觉，以为玩家的游戏数据全坏了，还上什么班啊，卷铺盖走人得了。之后的一两个月里，几乎一到周末就有某些区玩家进入不了游戏，由于有之前的案例，就再次添加索引玩家又能进入游戏了。奇怪的是，很过区服刚添加过索引不久，MongoDB 索引又丢失了，我们到处高手问，网上查找资料，为什么 MongoDB 索引会无缘无故就丢失呢，很多高手给的答案就是要么换掉 MongoDB 数据

库要么重新审查 MongoDB 的表结构的业务逻辑。当时我看官方文档说 MongoDB 是内存映射型数据库，我就怀疑是不是由于内存不够导致 MongoDB 数据丢失的情况，但是明明内存是够的啊。最后开发同事没法了就开始审核代码，最终发现游戏代码里面有后面程序，通过调用 PHP 的 eval()可以执行任意代码，再通过 MongoDB 的操作记录发现，MongoDB 索引丢失的时间段里，有很多删除操作。代码修复后，后面基本上就没有出现过类似事件了，他娘的，这不是要坑人吗，我们为此加了多少班，浪费了多少时间，为此，我决定以后有空了一定要对开发的上线代码进行关键字过滤。物理机房断电，物理机房调整防火墙，游戏域名没有备案被当局墙掉，公司 BI 系统遭受 CC 攻击等等类似的事情还有很多，一个人扛过来了，受益也颇深。刚来公司的时候，一边老板要让弄打印机，弄 RTX，另一边老大又要让处理线上问题，两头都为难。还有就是游戏频繁上线代码的问题严重得很，又要项目一直没有发布分支，导致很多时候测试刚测好，放到线上就又出问题了，原因是策划那边又改资源了，我这边也要频繁上线。刚开始通过 rz,sz 将开放打包的代码上传到服务器，后来实在受不了，就想法写了同步脚本，但是频繁登录到服务器去执行脚本，我又受不了最近研究 Rundeck，终于实现了在 WEB 界面去发布代码，后续会写文章讲解。

在这边工作，我的脑子每天都在高速运转，每天都在想如何才能减轻自己的工作量，如何才能提高工作效率，不让自己沉溺于繁琐的重复劳作。还有作为运维主管，如何去推动高效的运维，如何制定运维部内部员工的职业发展路线。我在大公司的时候我深切的体会到在一个公司做网管几乎就只能一直做网管，很难进行转岗 ,因为网管基本上只会 Windows 系统 ,但是就目前来看 ,学习 Windows 是没有太大前途的，除非想一辈子做网管。所以我在公司除了桌面系统推行使用 Windows 系统外，其他的内部服务全部使用 Linux 服务器，DNS,邮件服务器，域名代理等服务器，让公司的网管组同事也能够有一定的发展空间，除了常规的 Windows 桌面维护外，有很多学习 Linux 的机会，以后可以直接转岗到游戏运维。同时为了规范化运维工作，我在公司推动搭建公司内部的 WIKI 系统，无论是网管工作还是游戏运维工作都要记录到 WIKI 系统中，以让新入职的同事能够尽快熟悉本质工作，同事也鼓励公司内部知识分享，将一些工作经验分享到 WIKI 系统中，逐步完善公司内部的知识库。

2014 年我们需要做的工作还很多，去年经历的很多苦逼事件促使我们更快的成长，更快地寻找到适合自己发展的方向。主要有以下几个方面

第一，尽快完善代码上线流程。从过去的完全手动上传代码，到编写 shell 脚本，再到通过 WEB 方式去点击，总之，一切目的都是为了提高工作效率和避免重复劳动，运维工程师不应该被这些繁琐重复的劳动给拖累，应该去创造更多的价值，应该花更多时间去研究如何优化流程。
第二，完善监控系统。在过去由于时间精力有限，我只是使用 zabbix 初步搭建了一个监控系统，对于很多业务层面上的监控都没有做调整，如游戏域名的正常访问，MongoDB 监控等。

第三，所有的 Linux 服务器统一账号管理。在去年，由于情况特殊，项目开发具有服务器的所有权限，去年我晚上睡觉的时候都担心别人会误操作删除服务器数据，但是都是使用同一个账号登录服务器的，无法得知是谁登录服务器 ,所以今年我要仿照之前外企的账号管理模式 ,搭建 OpenLDAP 集中账号管理服务器，对不同人进行访问权限分类。
第四，对上线代码进行审核。在去年，由于前项目组程序员在游戏代码中留后门导致我们经常加班，经常是半夜打车回家的苦逼经历，在今后的代码上线之前一定要对开发的代码进行审核，避免由于有意无意的安全风险。

第五，和开发同事一起研究如何优化游戏架构。目前每个游戏区服都是使用单独的域名，单独的 Nginx 虚拟主机目录，然后同样的代码要拷贝多份，每个区服需要单独配置配置文件，由于程序的架构不支持负载均衡架构。这种方式太坑爹了，既不能合理利用服务器系统资源，运维这边也是干些重复的体力活。在后期我们会考虑使用 HAProxy+Keepalived 作为游戏服的前端 ,然后根据负载情况增加或减少后端的游戏服。这里需要考虑游戏玩家的 session 会话和应用日志处理的问题。

第六，深入学习 MongoDB 和 Redis。后面的项目，我们也使用 MongoDB 和 Redis 作为游戏的游戏数据库。所以，运维这边有必要深入了解 MongoDB 和 Redis 数据库。

第七，公司内部邮件服务器切换。由于公司是创业型公司，一直使用的是 QQ 的免费企业邮箱，随着公司的发展，无论是从企业的私密性要求和邮件的收发效率来讲，使用类似 QQ 企业邮箱这种免费邮箱会有很多限制和安全性隐患。数据放到自己公司内部才是最安全的，况且部门内部员工邮件沟通走内网也是相当快速的。在去年我们使用 iRedmail 开源邮件方案在公司内部测试，我平时除了正式的工作邮件交流使用公司的正式邮箱，其他的邮件全是使用测试邮箱账号进行收发邮件，如接受 zabbix 监控报警邮件，注册国外网站使用的邮箱，包括公司内部一些系统的通知邮箱地址，如 xwiki 系统，redmine，代码发布系统全是使用测试邮箱账号。从使用状况来看，iRedmail 开源邮件解决方案还是比较高效和稳定的，所以，今年我们会抽时间将邮件服务器从 QQ 免费企业邮箱迁移到 iRedmail 邮件服务器。

第八，推进自动化运维。工作第一年我花时间研究了 puppet，结果没有使用，现在也没有再去研究了，由于 puppet 是使用 ruby 语言写的，puppet 的配置语言和 ruby 也类似。我决定直接放弃 puppet，选用 SlatStack 作为我们的自动化运维工具，它由 python 编写，很方便进行二次开发，同时正在使用的自动化工具 Rundeck 也可以整合 SaltStack,所以，StaltStack 是今年我们需要研究的重点。

2014 年，是一个充满机遇和挑战的一年，我会更加努力努力去做好自己的本职工作，带领团队成员打造优秀的运维团队。

IT 人的自我导向型学习：学习的 1 个理念和 2 个心态

作者：周金根　来源：http://zhoujg.blog.51cto.com/1281471/1376239

写这一个系列之前，我定位是与高效学习有关，不写生活但是会涉及一点点生活，第一版不要求整个系列有严密逻辑但是每篇文章一定保持高内聚结构。总体方向确定，但是真的在我现在提笔开始写的时候，我却停了一下。学习这个话题太多了，之前也写过一些相关的文字，加上后面新的经验和感悟，我该从哪和大家说起呢？

前一阵子我在思考学习的时候，想起之前《个体执行力》主题演讲中讲到的一张图，索性就拿这张图和大家聊聊吧：）

会学习吗？

如果我说大部分人还不会学习，可能很多人心里不服，毕竟读书二十余载都是在学习，考上大学也表明你聪明，然而事实上我们的确很多人工作几年之后，仍旧掌握的是一些皮毛表层的东西。面试过很多人，工作五六年自称都是高手，做过的项目落满多张简历，然而最后发现也只是会拖拖控件，设计模式不懂、架构不懂、沟通协作不懂..... 我不是说每个人都要学这些，只是你这些都不学，只拖拖控件对你真的没有什么帮助的。

我有时候想，为什么五年了还只会拖控件？他们难道就不想成为技术高手吗？说到这，我需要替这些人说说话了，没有哪个技术人员不想站在几百人的大会场去给别人做演讲，没有不想成为高手的技术人员，只是我们的确需要体谅他们的辛苦，经常的加班早已拖累了疲惫的心，一两个月一个项目，自己只有重复写着差不多的代码，根本没有时间来学习。对，我说到“没有时间”和”重复拷贝代码“，说到这，我可要说说这些仍旧只会拖放控件、不懂设计模式的朋友了，有人说时间就像女人的乳沟总是挤出来的，虽然比喻看似有点上不了正台，不过时间这个东西时最公平的，每个人都一样是一天 24 小时，你怎么利用好它，它就怎么回报你。敏捷个人从不说自己没有时间，只有自己不愿意。再说重复拷贝代码吧。连你自己都知道这是没有意义的事情，为何总是不能停止这样的举动呢？我知道，你会说项目忙，不可能有这样的时间。时间真是可怜啊，说来说去，又拿时间来做借口。你从一个城市老远背井离乡，没有关系没有后台，又不会偷鸡摸狗，剩下的就只有靠自己努力了。学会长大，学会承受，要知道没有人是必须要帮你，你的项目经理不行，甚至你的公司都不行，只有你自己才能让自己独立、坚强、幸福，所以我们还是收回没有时间这个借口吧，好吗？

当然我并不是说要会学习就要先懂得时间管理，这也不是我想阐述的，因为学习这个范畴和时间管理属于两个交互的学科，除了时间，我们再说说目标吧。每年写年度计划时你什么心情？也许年轻的心砰砰跳了几下，心潮澎湃的制定了宏伟的学习计划，每周看一本书、学习几门语言、考高级程序员认证等等，然而你真的会做目标吗？你懂得如何保证这些目标会真的执行下去吗？如果执行，你又如何高效的做下去，别人做两天，你一天就够了，而且质量并不比别人差。

上面我一股没有逻辑的言论，说的和学习有关，但有不是学习本身，越聊对学习就会越迷茫，越不知道学习到底是什么，就像不知道”我是谁？“一样。你有这种感觉吗？
程序逻辑写多了，喜欢结构化思维，下面我还是先从理论知识的角度去理解一下学习吧。

学习力

也许看过我文章的人会说，前面的文字感觉和我以前的风格不一样啊。没错，我就是想换一种口吻和方式来写，换一种不同的方式来做相同的事情可以有更多灵感和兴趣。不过，我毕竟还是一个逻辑性思维的人，不可避免的大家还是会看到我的一些 PPT，当然我也知道大家还是比较喜欢看我的 PPT 的，只是内容文字简洁，但信息量却很大，不管你能否全盘接收吧，看过总会有印象，后续我还会就图中的内容继续聊、多次聊，你也不用担心一下理解不了。

我们先来看看下面这张图吧，敏友们是否有印象呢？

这个图是第一次出现，但三个维度的总体框架图早在执行力线下分享时讲到。不过估计很多人都不记得了，因为的确我每次分享的信息量太大，这里我要像大家道歉了，我是一个自私的人，我是以提升自我为先，再是影响他人，所以为了每次加深理解和学习更多内容，我就要求自己准备很多，从而给大家带来不便，真心对不起了。我的原因说完了，现在我们换个角度来看，如果你一点印象都没有，那我可能会问你一个问题：“你为什么呆了一个下午，对重要的内容会一点印象都没有？“其实，这就是一个学习的问

题，你可以说我讲课没重点，或者当时我说的对你来说不是重点，说什么其实都可以了，我只是举个示例，除了这个图之外，在你的工作中一定还有学了和没学一样的情况。其实，每个人都会忘记学到的东西，上面只是我快人快语，目的不是苛求，只是想唤醒大家去思考一个问题：”为什么有的东西你能学得很好，为什么有的基本就忘了呢？这其中有什么规律？“你想过这个问题吗，如果没有想过，你可能每次接到一个新任务，都需要重新摸索一套方法，这其实是非常不高效的。
其实我有时连我自己讲过的课、做过的 PPT 也会忘记，如果我苛求你们，那我接不能取纳自己了，学习其实是一个比较难的事情，首先是你要学什么？什么时候学？怎么学？在哪里学？.....

我意识到我好像说到 5W1H 了，5W1H 是很多人常用的实用工具。就这一个 What，就足以让我不知所措。学习是什么？很多看似简单的问题背后却包容大的话题。问题岁不好回答，不过我终究还是要为自己找到答案，否则我怎么认识自己呢。于是开始思考学习，才有了上面的这个图。图其实是一个概念和关系化知识的一个必经过程，以后会具体说这个，现在还是回答图本身上。我要申明一下，PPT 的图都是我原创的，但里面的内容却不是啊，如果你要问我内容来自哪里，我不是不愿告诉你，其实大家也知道我是非常开放的告诉大家想知道的任何东西，只是我真的已经不记得了，也许是哪本书的一段话、一幅图或者是哪篇 blog，总之都是被我偷来后被我加工的转手原创品。如果你觉得我的内容还行，那么很高兴我会继续与你分享我更多的加工物，如果你觉得我是一个捡垃圾的大杂烩，那也没办法，你就将就着继续看，说不到哪天你就喜欢上我的内容了。

既然我们说到内容了，那我就不罗嗦了，下面说说具体内容吧，也就是学习的 12345。这个 12345 是今天想出来的，其实就是为了写这篇文字，既然是给大家看的，我就要表现出一点水准，当然不能毫无逻辑，还欢迎大家指正，一起把学习这个事情稍微一点理论化，帮助更多人搞定职业者的学习。

一个学习理念

3 年前的敏捷个人就强调意识、方法和工具。其中以意识为先 ,这在大多数事情上也成了我做事的原则，不管学习什么，如果我能先了解理念以及我为什么要去学的话，那我将可以快速让自己投入进去，大家可要知道，专注给你带来的效率提升绝不容忽视。

我的知识比较狭窄，有时蛮羡慕那些随口就能侃到天文地理、政治经济、军事运动的人，知识面丰富。不过我知道的不多，反而可以让我更快的定位吧。关于学习，脑子马上浮现的是孔子的“学而时习之，不亦说乎？”，这也成了敏捷个人的学习理念，简单的说，就是学习是为了快乐，这也与敏捷个人价值观非常的吻合。

不过一定有人马上跳出来了，学习=快乐？疯了吧？学习=痛苦，还差不多。先不和你争论，没什么意思，我深知信念这个问题不是简单聊聊天就能改变的，我还是把我的观点先说一下，你再选择接收哪些吧。

孔子这个人我就不介绍了，其实真要介绍我也说不出啥，毕竟我人文知识比很多人都差，不过既然我觉得孔子的这句话听起来很舒服，那一定得去理解一下。原来以为孔子就是文人，后来看了写书才知道孔子除

了弹琴，还能驾车射箭呢，他做过很多工作，曾经还管过仓库、放过家畜，这些都对他的学习理念有了影响，孔子一生对诗、书、礼、乐、春秋、易进行了系统的学习、整理和研究，《论语》中有多处孔子指导学生学习典籍的记载；孔子不仅重视技能与典籍知识的传授，还很重视思想伦理道德与社会政治方面的教育，也就是注重知识、技能和态度三个维度的教育。上面说的不对不重要，重要的是我们对“学而时习之，不亦说乎？”的理解。

这里说的学，包括学的内容和学的途径，孔子教育弟子首先首先从生活实践中去学，）是善亦学，不善亦可学。事事可为师。向身边的人学，”三人行，必有我师也，择其善者而从之，其不善者而改之。”

这里说的习，可不是一般意义上的温习，而是复习、思考、研究，并有所新发现新收获的过程，否则你哪来的快乐啊？温故而知新，可以为师矣，把学过的东西教一遍，在教的过程中既温习了旧知识，而从中有所启发得到新收获。

前面我已经说了说到底就是快乐的学习，而快乐的重点是习，也就是你会有新收获的快乐。你可要知道，愉快学习可是提高学习效率的最佳手段啊，那种痛苦的填鸭式的学习真的不值得借鉴。

如果你问我为什么获得新知识就会快乐呢？人生的一大乐趣就是不断探索未知，当你知识越多，你就会发现你不知道的越多，这正如爱因斯坦说的：知识如同圆，已知的圆越大，其与外界所接触的无知便越多，这就越加激发起去了解和学习的兴趣，并从中产生无穷的乐趣。

观念说完了，理念的问题说简单就是你相信就有，不相信暂时拿你也没办法，更别说是我写篇博客给你看，也不存在什么利益关系，更不能强迫你去接受了，不过我真心希望大家真的去做到以学习为快乐，这对你来说只有好处没有坏处，一旦你有了这个理念，我们要做的就是后面的 2345 了。

两个学习心态

心态这个东西最不好讲，幸好我们这种分享的友好关系让你还能静下来继续看这个内容，如果换成是你不喜欢的公司领导同事对你说，你早就心里嘀咕着说个屁了。说真的，有些领导自己都是假惺惺的扮好心态，这又不是看不出来，不过我们对事不对人，即使坏领导说的那些要注意有好心态的话还是很正常的，这个我们可能否认啊。

关于心态，我们能讲很多，不过这个系列的主题是学习，那当然要讲学习了。幸好我的知识又不丰富，每个领域牢记几个重要概念并吸收成了我重要的学习方法。几年前和大家分享过《学习的心态》这本书，我觉得这里面有两个心态还是很有指导性的意义的，可以作为我们学习之处的原则，可以更好的认识自己的学习，或者说可以更好的接纳自己和持有正能量。
下面我们说一下第一个心态：**渐进累积**

你遇到无法解决的问题时，你会受到打击还是更加兴奋呢？你对未知知识和技能的把握是否有信心？你是否相信学习的力量？

你觉得你是天生不会读书的人吗？你觉得你天生就是不擅长工作的人吗？如果你相信人生下来就决定你后半辈子的生活，这就是持有整体理论者的观点。他们将成败归结与一种与生俱来、无法改变的能力水平，他们认为自己的综合智力货技能水平是一个固定的、无法继续演变的整体。

而对于渐进累积者来说，我们承认当前的不足，但这只是对当下的自己的评判，我们还有一种背后的力量在告诉自己，将来的我就不是这样的了。我们将成就更多来归功于长期的努力，人定胜天，世上无难事，只怕有心人，只要经过自己的努力，循序渐进，新手也能成为大师，我深信不已，你呢？

你是天赋决定论，还是终身学习论？前者是做能让自己发挥最大潜力的事，后者是把要做的事做好。天赋固然重要，不过我相信很多人其实更大的问题是如何做好当下的工作，你不就是这样的吗？我觉得当下的事情要做好，就要做一个会学习的人。要做会学习的人，首先你就要懂得学习是一个强调知识和经验积累的过程。人生而不同，选择做什么当然重要，但是不管做什么事情，做事的态度决定了你的积累过程是否有效的。我们不要每次遇到事情做的不好的时候就全归咎于自己在这方面有没有天赋，这样只会让你停留在粗浅的水平；我们也不要走另一种极端，为了让自己不放弃，一直相信自己是高手，然而当下又不是，于是不接纳自己，痛苦不堪。不管你以前是哪种人，我希望你现在开始去思考你是否愿意成为一个渐进累积的学习者。这种人注重过程胜过结果，遭遇挑战时会迎接挑战，遇到无法解决的难题时也不会让自信心受挫，这不就是一种幸福吗。

第一个心态说完了，接着说第二个吧：创造软区域

创造软区域

当你遇到低落时，你会及时调整过来吗？当你的环境不利于你学习成长时，你是如何逆境成长的？你能随时激励自己，给自己创造机会吗？

印度有一个寓言故事，说的是一个人想步行穿过大陆，但道路布满了荆棘，这时候他有两种选择，要不就是铺一条路去征服大自然，或只需给自己准备一双草鞋。如果是你，你会做出什么选择？我想大部分都会选择穿鞋，因为这个工作量最小。穿鞋是最有可能实现的方案，然而这只是故事的浅显含义，背后要说的是穿鞋是改变自己去适应环境，而想铺路是去改变大自然，一个从自身出发，而另一个是从而外部环境出发。你想想，工作中你是否发过什么牢骚，这是从自身出发的，还是对外而发的，这种牢骚反过来是不是影响了你去积极的工作，而你又深知不正面积极的工作只会给你带来默默无为。

硬区域是要求有一个合作的空间让自己正常工作，对应故事的铺路。软区域是心平气和的对待事情，适应它并学会使用它，对应故事的草鞋。经常有人说，学习一定要找个安静的地方，当然我们说有些事情一定需要安静的地方，然而如果我们一味的一定要追求做事必须有一定的环境才开始的话，那就是陷入了

硬区域的控制。硬区域是一个玻璃罩，它故然安全，不过太脆弱，而一但碎了，环境变化很大，罩里的人可能不堪忍受。而处在软区域中的人就像海里的海带，在海里顺着海流飘动而没有被海流冲断。

每个人都想成为架构师、CTO，但是你的工作要求、周围环境、领导者等都有可能是你不能现在选择的，这时也面临着硬区域还是软区域的选择，我们应该做得是认识到“软区域”的重要性，不能指望周围给你提供的都是你理想的，如果我们还想追求卓越，那么我们就必须适应自我激励和控制的生活方式，从容应对所发生的一切，像呼吸一样自如。这种“软区域”的应对方法比单纯的拒绝它更有效，它不是与生俱来的，而是需要培养的，以下为培养“软区域”的三个步骤：

1. 学会**平静的对待**生活中的不完美之处，适应自己的情绪，了解如何让它们自然宣泄出去
2. 学习如何把不完美的地方**转换**成我们的优势，激发我们的创造力
3. **自我激励**，不管外部条件是否有激励性，找到一种**激发最佳状态的情绪**，学习如何在我们的意识中制造一些波动来激励我们前进

不知不觉已经 0:31 了，虽然还不是很困，不过我还是要保重身体，今天就说到这里，学习的 345 下次再说

IT 人，身在北上广，你做到了这些么？

作者：lovelace521 来源：http://lovelace.blog.51cto.com/1028430/1375578

开篇从一个不算笑话的笑话说起：


俊义
@俊义和追马
追马
我现在做运维，纯属打杂的...
追马
俊义
“运维”听起来好牛逼的样子
俊义
又是无数个挑灯奋战的不眠之夜，一年后终成一运维...
恭喜你额...
追马
俊义
追马，你现在做啥工作的，看你天天不停的补....
我现在做云计算，纯属打杂的...
追马
俊义
“云计算”什么掉东西，云里雾里的.....

上图仅供娱乐，信息量略大，看的懂得如被戳中要害，请勿大喜大悲，看不懂的请绕道而行，旁边的道路上有美女，美女在哪里？

开始我们这篇博客的主题,事情的起源是这样的，话说追马这几天在找工作，不小心碰到了一个高大上的面试官，然后面试官生生的给追马上了一堂课，一堂关于 IT 人如何在北上广活下去或者是如何活的更好的课.....

面试经历如下图所示：


俊义
@俊义和追马
之面试
追马
俊义
这位童鞋，请介绍下你自己....
我叫追马，未婚，工作经验（此处省略一千字..）
追马
这就是我的情况....
追马
俊义
关于互联网动向你了解那些？
这么简单的问题还来问我，哪个腾讯不是把点评给收了么.....
追马
滔滔不绝的....（此处省略五千字..）
追马
俊义
你知道么，你说的这些就像娱乐圈的花边新闻，这不是你做IT该关注的.....


俊义
@俊义和追马
之面试
追马
啥玩意，那我该关注啥？
追马
好歹我也是过来人，还要你来教训.....
追马
俊义
你知道intel正在做万兆网+haddop+ssd向小型机方向转型.....市场上postgresql是在吃oracle的剩下的骨头.....
好吧，这些问题还真的没有关注过...
追马
俊义
你平时有参与什么社团活动么？
很少参加...
追马

从上图我们可以看出，面试官问的问题不单单局限于技术这块，而且还考虑到了作为一个技术在上海这个地方有没有一个扑捉 IT 风向的敏捷思维以及是否参与到了当地相关团体社区中去。

由此我们可以总结出三个结论：

1、在北上广三地做 IT，技术是基础，是硬性的指标，古语说得好：“在绝对的实力面前，所谓的技巧真的是不堪一击”。所以要想扎根做技术，“功夫”一定要过硬，否则永无出头之日。

2、关注业内动向 扑捉无线商机，不是那种花边新闻 而是实实在在的业界动向

3、参与到当地的社区和线下组织当中 争取和他们的发起者成为朋友 拓展自己在圈子内的人脉资源

做到以上三点，基本上说你可以靠技术吃饭了，但是想要发财，那就……. 原因你懂的…..

废话说了一箩筐，画图画的很操蛋，其实也没说什么有用的东西，只希望给已经从事 IT 行业的或者将要从事这个行业的人说一声，你要学习的不单单是技术，而你要拓展的也不仅仅是自身，

你要以自己为圆心，向外发散，当你真的做到了可以靠自己来影响周围的人亦或是影响这个圈子的人亦或是成为了这个圈子的领跑者，那么恭喜你，你已经名就了，可以依托于此向功成"进军”

共勉吧：

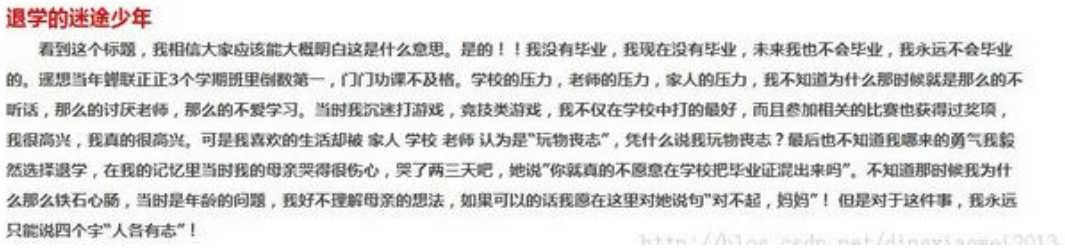
”天下风云出我辈，一入 IT 岁月催。

整合并购谈笑中，不胜人生搏一回 “

同是 90 后，努力需更多！——自我反思和小结

作者：丁小未　来源：http://dingxiaowei.blog.51cto.com/4561335/1379622

吃完晚饭，我还像往常一样，来到办公室，继续今天的工作和学习。这两天，我的点情有点复杂，有喜也有悲。可能是以为工作的原因，昨天还有今天，一直重复着性的操作，就是将 Excel 表格中的数据整理插入到数据库中，整个人完全没有了思想，只是一味的盯着屏幕，满眼里都是数据，时常都有插入混乱的可能。可能这是一份轻松的工作，不需要动脑，就简单的动动鼠标，敲敲键盘，我终于体会到有时候，别人眼里轻松的工作，自己实践起来并不轻松。何为一份好工作？何为度过充实的一天？有的时候感觉很苦恼，感觉过的很压抑，不是因为工作有多压力，而是感觉没学到什么东西，之前碰到些问题，感觉很苦恼，都曾发表过感慨，觉得自己这么菜，还能胜任程序员这一岗位吗？结果遭到了朋友们的“围攻”，或者不是一个好的程序员的一个心态，我们应该自信，做任何事都要对自己有信心，如果自己都对自己失去了信心，那何谈人家去相信信任你！可能是因为人的贪婪，人都是不满足的，当拥有很多还期望这更多，但求知的过程也是如此，我们每个人都希望能成为 IT 界的大牛，成为一个让人家敬仰的人物，不谈做到 IT 界文明，起码要能达到某个专业领域文明的人。接触了 Unity 这四个月以来，让我也了解了雨凇 Momo，想必 Unity 圈子里大家都认识他，在这个开发圈子里，他已经成为我们公认的大牛，可以说他也已经实现了他的价值，但或许他想要的还不仅仅是我们所认为的这些，他还在继续努力着，朝着他更高的理想在奋斗。我有一次，偶然进入他的空间，让我更了解到了他的过去，但让我惊讶的是他之前也是一个”顽固不化“叛逆者。



退学的迷途少年

看到这个标题，我相信大家应该能大概明白这是什么意思。是的！！我没有毕业，我现在没有毕业，未来我也不会毕业，我永远不会毕业的。遥想当年鲤联正正3个学期班里倒数第一，门门功课不及格。学校的压力，老师的压力，家人的压力，我不知道为什么那时候就是那么的不听话，那么的讨厌老师，那么的不爱学习。当时我沉迷打游戏，竞技类游戏，我不仅在学校中打的最好，而且参加相关的比赛也获得过奖项，我很高兴，我真的很高兴。可是我喜欢的生活却被 家人 学校 老师 认为是"玩物丧志"，凭什么说我玩物丧志？最后也不知道我哪来的勇气我毅然选择退学，在我的记忆里当时我的母亲哭得很伤心，哭了两三天吧，她说"你就真的不愿意在学校把毕业证混出来吗"。不知道那时候我为什么那么铁石心肠，当时是年龄的问题，我好不理解母亲的思法，如果可以的话我愿在这里对她说句"对不起，妈妈"！但是对于这件事，我永远只能说四个字"人各有志"！

当我看到这段话的时候，心里一怔，或许当初老师同学对对他感觉很失望，好好的一个孩子父母培养到了这么大，怎么可以如此的叛逆，但我又欣赏他好强的个性，他很清楚对自己的定位，正如他所说的“人各有志”，中国式的教育，我个人感觉有些死板，基本都是让孩子走的一个模式，我想每个人都有每个人的特点和长处，我们不能要求每个孩子都学习得很好，各个方面都很优秀，但从雨凇 Momo 的这段经历中，我感受到，人生没有失败一说，每个人都会有成功的机会，就看你是否够努力能够争取到他!最近我有看过 CSDN 上好多好多人的博客，在我们眼里都是一些比较有名气的人，发现他们都有共同的特点，就是能够沉住心，都有一段沉淀的过程，想必我们都听过飞蛾和人的故事，不经历破蛹之痛的飞蛾是没有生命力的，是的，对于我们学习 IT 技术也是一样，对于新手在工作初会碰到很多很多问题，是时候我们也会抱怨，为什么其他人也是这样做的他就没有问题，我这样就有问题，所谓困难就像弹簧，你弱它就强，你强它就弱，说明我们的“内力”还不够，基本功还不够扎实，有时候不懂得其中的原理，只会看到表现，貌似咋一看像理解了，其实并不然，新手有的时候重要的一个学习方式就是看视频，但看视频有一个弊端，就是让人容易满足，感觉自己一看，看老师讲解的很顺利，自己也以为“理解”了，其实不然，这里的理解要加引号，没有经过亲自自己的动手，别人再怎么讲还是他的理解，你只是按照老师的思路看懂了而已，仅仅是看懂！最近我也为考研没考上的同学们感到很遗憾，辛辛苦苦的努力，起早贪黑，辛苦到头来却回到了起

点，如果说，这整整一年的复习时间，如果你看重是每一天的过程，而不是仅仅在意的是最后的结果，我想最后并不会这样，没考上的同学，有谁认为自己复习的阶段过得很充实的，每一天都感觉很有收获，如果你做到了这样，想必经过了 365 天的沉淀，就算最后的结果不如人意，那又如何，起码你收获到的是一种好的习惯。在学习的过程中，如果仅仅是动眼，而不动手和脑，这等于是练拳不练功，到头一场空！
在看博客的过程中也认识了好多人，其中就有一个叫小雷的，我昨晚也跟他 QQ 聊过，看到他的博客，了解到他的过去，似乎也跟雨淞 Momo 有相似之处，也是一个大家眼中的“叛逆”着，他曾在学校两次因逃课被处分，但他最后还是坚持了自己的梦想，自己专研，自己动手，成为了我们所公认的大牛！我所理解的大学，除了顶尖的 985，我感觉如果是按照一般学校的培养模式，发展不会很大，甚至找工作都成问题，其实大学计算机专业，这应该是一个新兴高科技的专业，毕业之后，有可能拿到上万 offer 的，也有可能找工作都成困难的，绝大多是是后面这种，现在大学生的学习和就业的现状，相比我们也都很清楚！有的老师眼里的好学生，遵纪守法，学习考试成绩优异，但这我感觉并不太适合 IT 的学生，可能是我个人的理解（是对是错，不喜勿喷），有的好学生年年国家奖学金或者是优秀学生干部，但到最后毕业设计还是一点都不会，感觉除了多认识了人，多解除了到几门课，到毕业仅仅还有印象的可能就剩下学科的名称了，感觉也没学到什么！计算机专业的学习不是看出来的，也不是听老师讲课听出来的，而是自己多动手写出来的，总结出来的！这个我一直崇拜的偶像杨中科老师说的，当然也是跟我之前所理解的是契合的！
我感觉学习的过程中要善于反思和总结自己的得与失，这样才不会过的很迷茫，在学习的过程要注重细节，当然写这些不仅仅是给大家看，更重要的是给我自己看，要戒骄戒躁，要沉淀，做到学习，永不止步，这就是我的个人名言，希望每天都能有新的收获。

大多数的新手，像我之前也是一样，总是想问人，IT 技术那么多，学什么好，学什么将来能够拿到更多的 Offer，然后有的同学就是今天听这个人说 Java 好，然后就下定决心去学 java，明天又听说.net 好，然后又半途而废的开始了新的 HelloWorld 征程，这样一段时间过后，感觉学的还是 Helloworld，不一样的仅仅是“外表”，或许是从 Java 的 HelloWorld 跳到了 C#的 HelloWorld，又或者是其他！其实技术本身并没有什么优劣之分，没有所谓的那种技术好，那种技术不好，就算你学的是过时的语言，但你如果养成了自己的一种思想，一种学习方法，那你就是大牛！大牛是如何炼成的，就是在沉淀中炼成的!为啥武侠小说中厉害的人物都有闭关一说，闭关就是就是摒除杂念，好好的静下心来，去提升自己的内在修为，而不是沉迷于花拳绣腿!高手就算手中无剑，照样能摘也飞花，就像扫地神僧，内力修为达到一定境界，就算是用眼睛都能杀人！我们学程序也是一样，做一个东西重要的不是用的学什么开发工具，在什么平台上开发的，重要的还是思想，学习不能仅仅是看，看的永远是人家的招式，而没能转化为自己的思想！也或许也就是张三丰在教张无忌太极剑的时候，问的一个问题，你记得多少了？而张无忌回答是忘了差不多了的原因!有同学获取还会拿这句话当做笑话来调侃，但有多少看懂了无忌话中的隐含的真谛呢!

新手还总是喜欢问，有没有有什么学习捷径，好多培训机构都鼓吹说几个月甚至几周精通一门技术！这完全是吹牛的鬼话！！！但就是能骗骗那些贪图捷径的“愚蠢”的人！花大价钱，想缩短学习时间来达到成为高手的目的！这完全是吃人说梦！！！学习其实是没有捷径可循，在来第一家公司也有四个多月了，

每天都睡得比较晚，不知道多少个夜晚都两三点，印象中很少有在一点半之前睡觉的吧！感觉要学的真心好多好多，大牛是这样炼成的！还是我所说的沉淀！那我就让大牛们告诉你成为大牛的捷径是什么

李华明 Himi(CTO,90 后 移动开发专家) ,这些词都让我们羞愧 ,为什么同样是 90 后 ,差别怎么就这么大？！

1 基本信息 编辑

姓名：李华明

昵称：xiaominghimi

英文名：Himi

出生日期：1990年1月1日

职业：CTO，青年作家

名言：不要让任何事情成为你不去学习的理由。

移动开发专家，专注于移动开发领域，多年 J2me、Android、iOS 平台游戏与软件开发经验；

2010 年年初与清华出版社签约创作《Android游戏编程之从零开始》一书，此书于2011 年成功面市，受到业界人士的关注与支持；

2012 年年初与清华出版社签约创作《iOS游戏编程之从零开始—Cocos2d-x与cocos2d引擎游戏开发》一书，此书于2013 年成功面市，再次受到业界人士的赞赏与支持；

在CSDN、ITeye、51CTO、eoe-Android、中国移动开发者社区、微度网等多家技术论坛担任技术专家与版主；博客中发表了一系列Android 与iOS 游戏开发的文章受到广泛关注！其中最值得提起的是博客中《浅谈3 年游戏开发de 自学历程！》一篇自述让众多的大学生与IT 业界人士大为鼓舞与赞赏；

李华明 [1]

我们在感觉到羞愧的同时，但不要放弃，我们虽然输在了起点，但不代表是终点！人生没有终点，学无止境，学海无涯！永远不要感觉到满足！我也不要觉得跟这些大牛们的差距遥不可及，只要努力，只要做到问心无愧！多沉淀，在沉淀中爆发！我们都会有出关的一天！

李华明的学习捷径

上班之前：

学习J2me的时候，每天除了饿了去吃饭之外，全部用来学习，基本上是3天里有一天通宵；
写了四款基于Kjava（J2me）的游戏、益智、飞行射击、趣味、RPG;
然后带着四款游戏项目面试上班；

上班之后：

在公司：

在公司做项目，有过连续2天1夜不睡觉，一周不回家的经历。经常通宵做项目很正常；

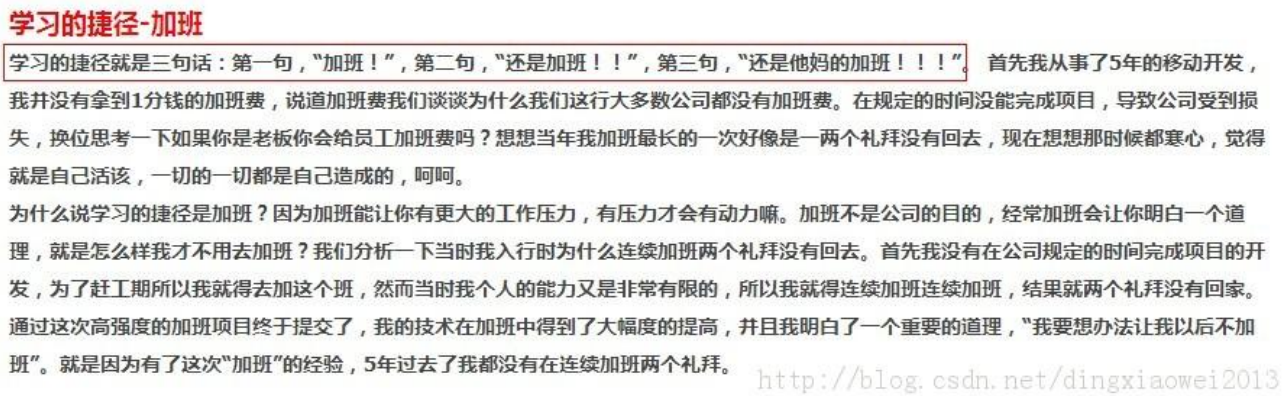
在家：

每天学习到深夜，习惯于凌晨3-4点睡觉，早上8.30左右起床上班；

雨凇 Momo(Unity 开发大神)，其实他的年龄也是跟我们差不多大，我具体不记得他是不是 90 后，反正也差不多！我原来以为他是大叔级别的大神，结果看了他的空间了解他的个人信息，我又羞愧了，还是

那句话，为啥同样是"90"后，怎么差距这么大？！也许我们在睡觉的时候，人家还在编码！这就是无形中拉开差距的地方！

Momo 告诉我们学习的捷径：



学习的捷径-加班

学习的捷径就是三句话：第一句，“加班！”，第二句，“还是加班！！”，第三句，“还是他妈的加班！！！”。首先我从事了5年的移动开发，我并没有拿到1分钱的加班费，说道加班费我们谈谈为什么我们这行大多数公司都没有加班费。在规定的时间没能完成项目，导致公司受到损失，换位思考一下如果你是老板你会给员工加班费吗？想想当年我加班最长的一次好像是一两个礼拜没有回去，现在想想那时候都寒心，觉得就是自己活该，一切的一切都是自己造成的，呵呵。

为什么说学习的捷径是加班？因为加班能让你有更大的工作压力，有压力才会有动力嘛。加班不是公司的目的，经常加班会让你明白一个道理，就是怎么样我才不用去加班？我们分析一下当时我入行时为什么连续加班两个礼拜没有回去。首先我没有在公司规定的时间完成项目的开发，为了赶工期所以我就得去加这个班，然而当时我个人的能力又是非常有限的，所以我就得连续加班连续加班，结果就两个礼拜没有回家。通过这次高强度的加班项目终于提交了，我的技术在加班中得到了大幅度的提高，并且我明白了一个重要的道理，“我要想办法让我以后不加班”。就是因为有了这次“加班”的经验，5年过去了我都没有在连续加班两个礼拜。

http://blog.csdn.net/dingxiaowei2013

搞 IT，没有经过那么多时间的沉淀和自我反省，是不会成为大牛的,不会有人是天才！我妈也一直“怪”我说，上班又不是上学，为啥整天把自己搞的那么累，你应该向你表弟一样，学习的多轻松！我想说，或许我比较笨，感觉累不是一天睡觉的多少，不是身体有多累，而是如果没一天没有学习到新的东西，一天没有什么收获，我感觉是特别压抑和特别累的时候，心累永远比身体累要累的程度要多得多！相信那些整天无所事事的人，问他获得怎么样，我相信回答也会感觉“累” ，并不会觉得幸福！

这一周，每晚都睡的特别晚，当然也不是故意这样，头两天，我觉得很好，虽然比较晚，因为每天晚上都坚持写下一天的学习博客，虽然都写到两三点，但觉得很开心，睡在床上都会特别的舒服！这是一种心理成就感！晚上花一段时间总结和回顾一天的过程，然后整理成文，既是一个温习，又是一个鼓励！我在博客的世界中，认识了好多人，当然也有好多人认识了我！有跟我们一样大的，他在 51CTO 工作，51CTO 相信 IT 人都会比较熟悉，跟 CSDN 是类似的！龙龙同学，他是北京科技大学的高材生，人很好，平易近人，很有哥们义气，帮了我不少忙，没有名牌学校的那种高傲，非常的随和，他让我有继续创作的信心！也有比我小的，国华妹，虽然才是大二，虽然我也不认识她，也是因为今天我跟她聊过之后，然后更了解了她，我想象中一个可爱甜美的学妹。她让我了解到他所在的学校，虽然是一个普通的二本学校，或许好多人都不认识，但他们学校又一个知名的 IT 教育的老师，米老师，我从他们这些学生的博客的字里行间，都能感受到他们对米老师那种尊敬和爱戴，米老师带出来的好多师哥师姐们毕业出来都能拿到上万的 Offer，知名的 CSDN 高校 IT 教师，也是教育行业大家所敬仰的老师——贺立坚老师也在博客中对米老师大为赞赏，看他们两博客中的对话，感觉这两位老师我认为是所有高校老师的一个楷模，他们把学生都当做想自己的孩子，学生们也把他都当做是父亲一样，感觉他们的班级更像是一个温馨和睦的大家庭，里面充满了友情和正能量！跟丁国华小学妹聊了一小会儿，但感觉很投缘，可能我们姓一样，但或许都是好学上进才会这么投机，看她的博客，我感觉很舒畅，写的非常的优美，非常有文采，我都幻想着这是一个怎么样的小姑娘，或许是以为环境，在那个大集体里，每个人都很上进，我感觉他们那个班级，不差于名牌大学里的学习气氛，说实话我非常的羡慕，有那么一个好的老师带领着他们启蒙，有那么一个好的环境，让他们成长！看

到他们能写出那么“优美”的博文，我顿时又感觉到惭愧，向他们学习！多总结，多反思，过好充实每一天！多问问自己，今天，我学习了吗？！

让我开心的是，清华大学出版社图书编辑找到我，鼓励我能够出自己的书，这让我更有信心继续坚持下去！我希望也能像 Momo 和 Himi 那样，能够出版自己的书！记录下自己的一个学习笔录！

skyyan夏毓彦 10:55:00
hi，您好丁小未
謓鈊黓銥 10:55:00
您好，我现在有事不在，一会再和您联系。
謓鈊黓銥 10:55:04
hi
謓鈊黓銥 10:55:06
请问你是
skyyan夏毓彦 10:55:07
我是清华大学出版社编辑
skyyan夏毓彦 10:55:14
夏毓彦
謓鈊黓銥 10:55:37
恩
謓鈊黓銥 10:55:46
你好
skyyan夏毓彦 10:55:54
呵呵，请问您那个 Unity3D博客，可以继续写成一本书吗？
謓鈊黓銥 10:56:21
恩
謓鈊黓銥 10:56:23
可以
謓鈊黓銥 10:56:44
我还在继续更新中
skyyan夏毓彦 10:56:46
嗯，我出过李华明的书。
skyyan夏毓彦 10:57:13
也是博客写着写着，内容多了就成书了
謓鈊黓銥 10:57:32
恩
skyyan夏毓彦 10:58:00
好，丁老师，我先联系上。
謓鈊黓銥 10:58:34
受宠若惊 不要这么称呼 直接叫小丁就好了
skyyan夏毓彦 10:58:53
出版图示关键流程比较简单，就是与出版社签合同，稿件整理好叫稿就可以了，后续出版工作有出版社完成
skyyan夏毓彦 10:59:08
叫稿-》交稿
skyyan夏毓彦 10:59:29
没有关系，有一技之长都是老师
謓鈊黓銥 10:59:29
恩 好的 日后如果整理成册 我再联系您好吧
skyyan夏毓彦 10:59:42
好，保持联系
謓鈊黓銥 10:59:50
你出版的是李华明的什么书呢
謓鈊黓銥 10:59:57
cocos2dx？
skyyan夏毓彦 11:00:19
夏毓彦

清华大学出版社图格事业部
EMAIL:booksaga@163.com
电话：010-82728184-803
手机：13701352995
QQ: 49659450
地址：北京海淀区西王庄5号楼305室
（清华同方大厦对面小区门口的宾馆3楼，欢迎作者及代理人直接到我处商谈）
网站：http://www.tup.com.cn/
skyyan夏毓彦 11:00:33
他的两本书都是我出版的
skyyan夏毓彦 11:01:08
他的书版权输出台湾也是我们做的
謓鈊黓銥 11:01:38
恩 清华大学出版社 挺不错的 出版的书籍我用的也比较多
skyyan夏毓彦 11:01:57
呵呵，很好。

在这聊天过程中，没想到到原来李华明 Himi 出版的两本书是也夏老师出版的，更没想到李华明竟然就比我大一岁，万万没想到李华明在 22 岁的时候就已经出版了自己的第一门书。只要心中有梦，就朝着她去努力，没有什么不可能，只有你想还是不想！

在新的一年里，做督促自己做到以下几点：

1.多看书！

2.多总结！

3.多反思！

以上是我跟陌生的国华小学妹聊天的感悟，我相信她日后也会发展的更好！以上的感悟我也是自勉，放下包袱，静心沉淀，学习，永不止步！

网站备案那些事----云里雾里知多少？

作者：lovelace521 来源：http://lovelace.blog.51cto.com/1028430/1380746

前言：备案流程几大步：

文章目录：

1、购买域名

1.1、国内外几个域名提供商推荐

国内：

万网： http://www.net.cn/ 推荐指数： 

西部数码：http://www.west263.cn/ 推荐指数： 

时代互联：http://www.now.net.cn 推荐指数： 

国外：

godaddy http://www.godaddy.com/ 推荐指数： 

国内外域名详情请查看：http://www.it.com.cn/f/server/063/31/253322.htm

http://www.chinaz.com/web/2009/1214/101041.shtml

1.2、域名购买时需要提交的信息

域名注册人信息填写，域名管理人信息填写

Note:域名所有人信息一经填写修改起来比较麻烦，需要注意，其他在域名后台管理页面 即可修改。

1.3、域名购买后需要提交哪些材料

个人：身份证正反扫描件

企业：组织机构代码证或企业营业执照扫描件

2、确定网站文件存储空间是购买还是自备

2.1、网站文件存储空间购买的话该如何备案

如果你网站文档存储空间也是购买的话，可以委托供应商代为办理。这种情况一般是购买域名的同时购买空间，比较方便。

2.2、网站文件存储空间为自己提供的话该如何备案

如果网站文档存放空间是自己的服务器的话，这个时候供应商就不会鸟你了，你需要自行解决备案问题。解决方式：

自己准备材料去备案，周期大概为 25 个工作日左右。无费用产生。

委托第三方办理，周期大概 7 个工作日左右，费用 200+（RMB）,这个第三方会看你域名的价值而要价的，不是统一的标准。

这几种方式我们等下将会详细讲解。

3、备案的流程

3.1、我们这里主要讨论服务器是自己的情况下，而且是自己自行备案的企业域名情况。

先决条件：

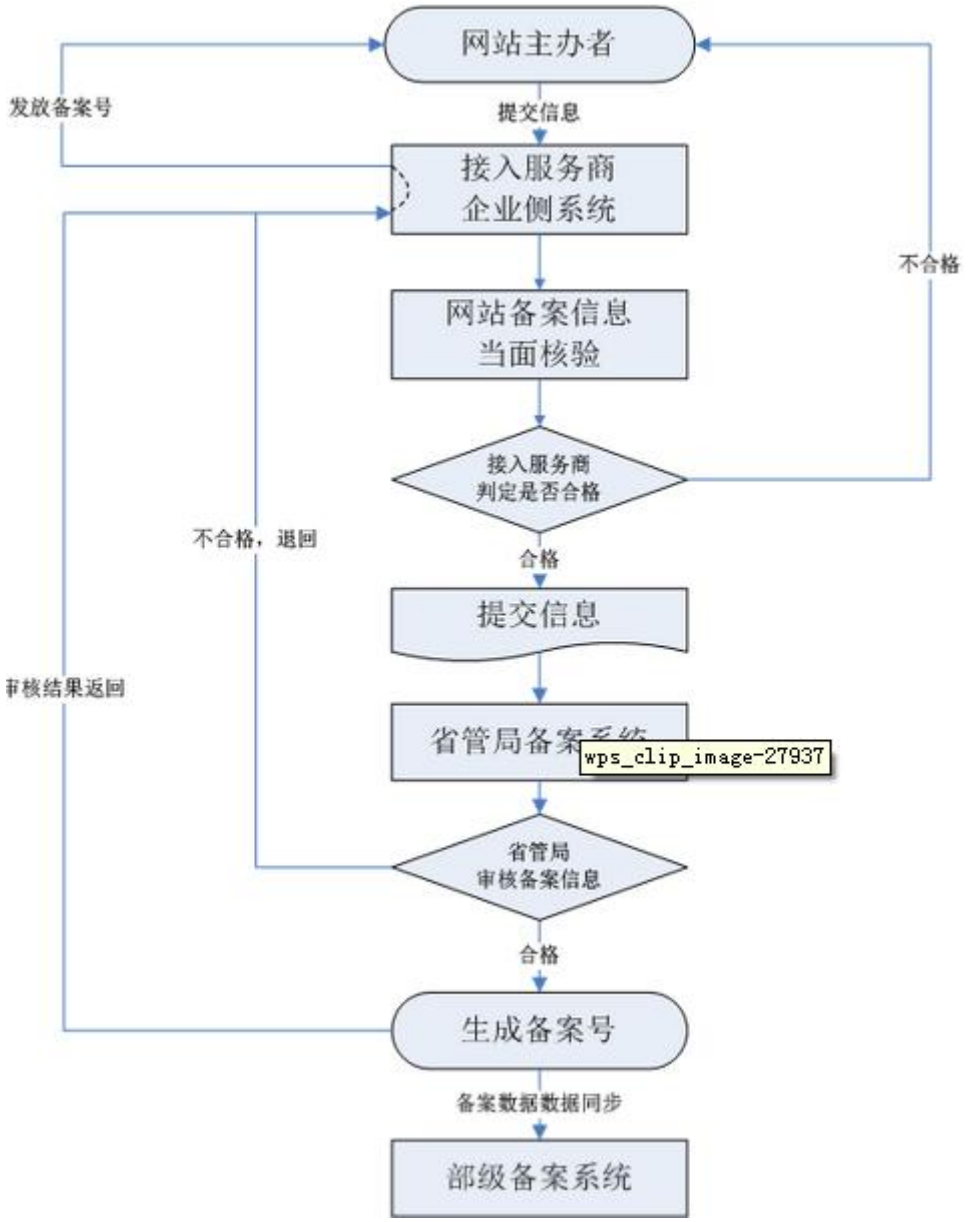
一台存放网站文档的服务器

一个固定的公网 ip

一个已经注册好的域名

Note:在天朝，哪怕你只有一个 IP,没有域名，要是放到公网上，那也必须备案！

3.2、官方给出的 ICP 备案流程



3.2、查找所属地区的接入服务商企业侧系统入口

这里提供三大 ISP 供应商的企业侧系统入口：

中国电信：http://beian.ct10000.com/portal/

中国移动：

http://beian.chinamobile.com/user/login.jhtml;jsessionid=A9131C589790032F3DA2AC5F169318A8

中国联通：http://124.65.49.12/filsys/security/login.jsp

上海通信管理局备案网站：http://shca.021beian.cn/

中国工信部备案网站：

http://www.miibeian.gov.cn/state/outPortal/loginPortal.action;jsessionid=3GcmTqyZsyqGv4LZ8MW5g1mdNFdDcrc6nvLGsmyL2Nnnyn6VpMfT!-1223402419

选择与你企业 ISP 对应的任意一个都可以完成备案信息提交工作，当然你也可以选择工信部的备案网站来提交相关信息，我的经验是最终还是要打回所在地的供应商的企业侧系统入口来提交相关信息，所以为了方便起见我还是建议直接在 ISP 供应商的企业侧系统来进行信息提交。

3.3、提交相关信息坐等接入服务商通知（信息无误的话周期大概为 3~5 个工作日内）

域名备案提交材料

当前位置：查询统计服务 -> 备案进度查询

备案进度

操作时间	操作人姓名	操作类别	操作结果	操作意见
2013-08-08	[redacted]	提交核实	通过	提交核实成功

核实成功之后，会往你留下的邮箱内投递一封邮件，内容就是要你准备相关材料在某段时间内到某个地方去办理相关手续。

（四） 网站主办者、域名所有者、IP 地址租用者须保持一致。

三、 核验场所地址和联系电话

地址：上海市南紫明路 1 号甲 2 楼 208 室（近四川北路天潼路）。

联系电话：63073654 63073473

工作时间：工作日 9:00-11:00，13:30-16:00

交通方式：地铁 10 号线天潼路站，公交：167、220、21、939 路等。

中国电信上海公司不良信息处理中心

2013 年 8 月 28 日

3.4、办理人携带公司相关证件材料前往目的地办理相关手续

3.4.1、网站备案信息真实性核验单。

3.4.2、单位工商营业执照（加盖公章）

3.4.3、单位网站负责人身份证复印件

3.4.4、接入服务商现场采集的单位网站负责人照片

3.4.5、单位主体负责人身份证复印件

3.4.6、网站所使用的独立域名注册证书复印件（加盖公章）

3.6、等待发放备案号（周期大约为 20 个工作日）

★ 关于ICP备案申请审核通过的通知

beian 发送至 nick；

尊敬的用户[redacted]，您的ICP备案申请已通过审核，备案/许可证编号为：[redacted]，备案密码为：[redacted]，审核通过日期：2013-10-11 15:30:40。

请牢记和妥善保管您的备案号、备案密码，备案密码将是以后变更备案信息的重要依据。

工业和信息化部网站备案系统

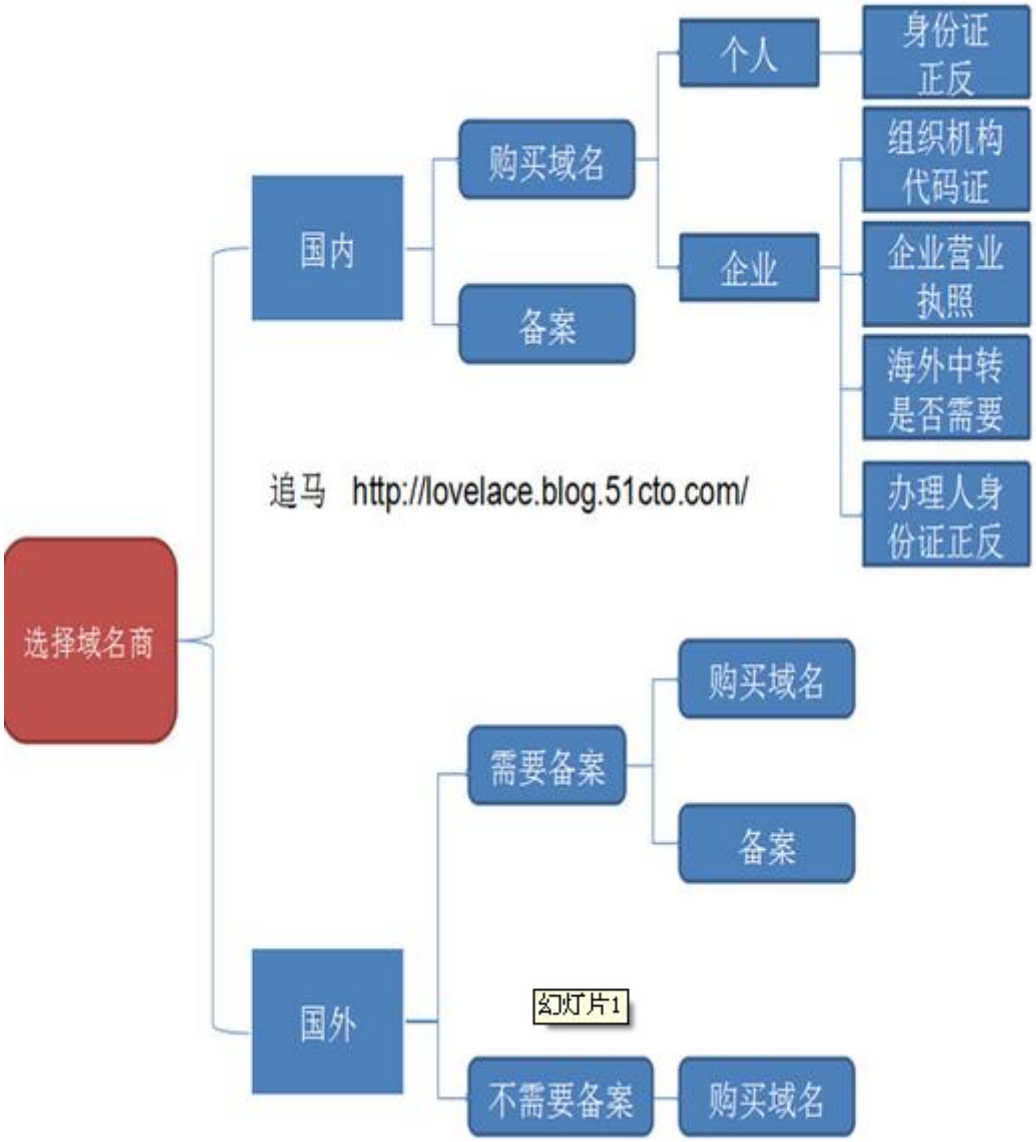
3.7、下载并导入证书

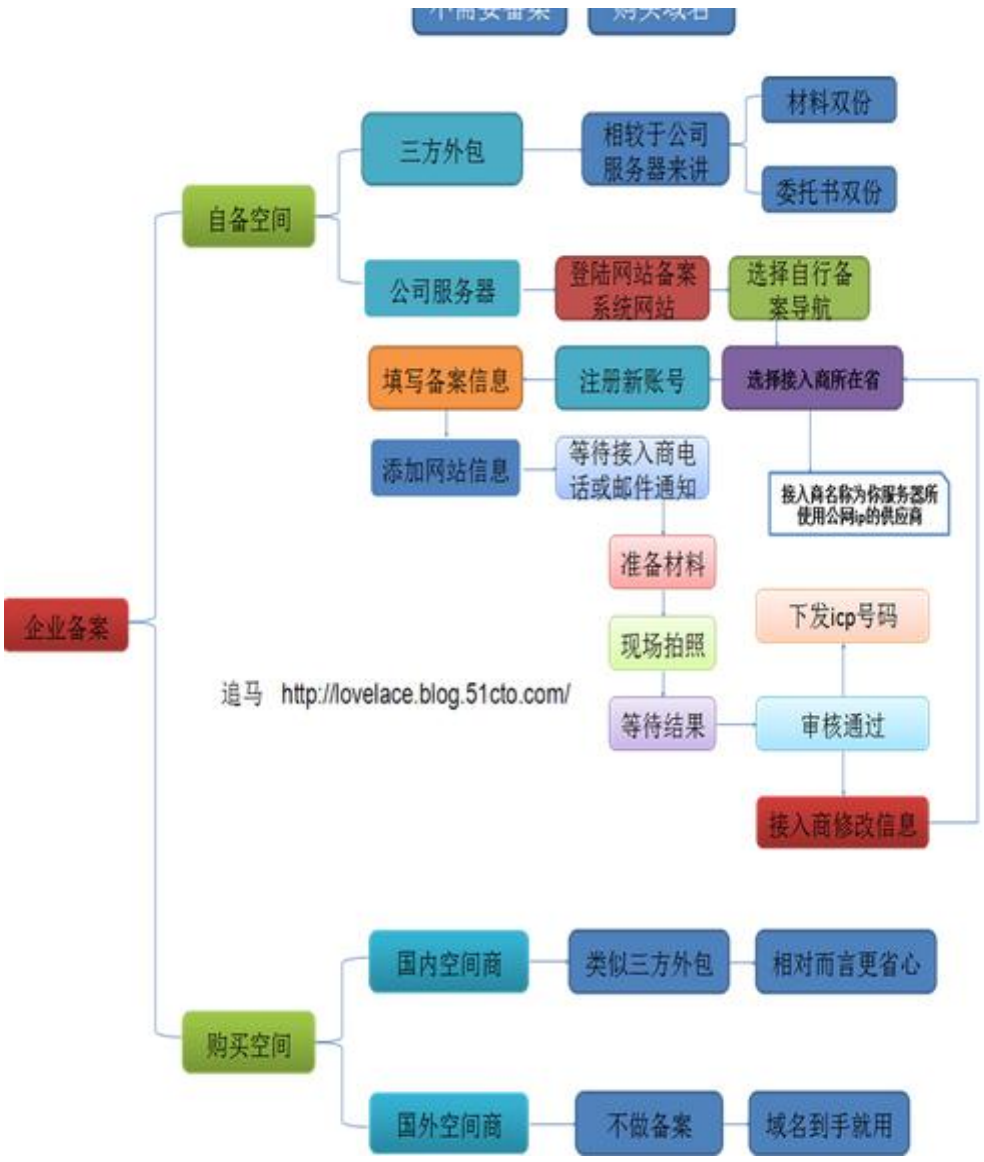
备案号获取之后，登录到所在地的备案管理系统或者中国工信部的备案管理系统下载证书，然后把证书导入自己的网站即可。并根据要就在网站首页底部中间位置显示备案号。

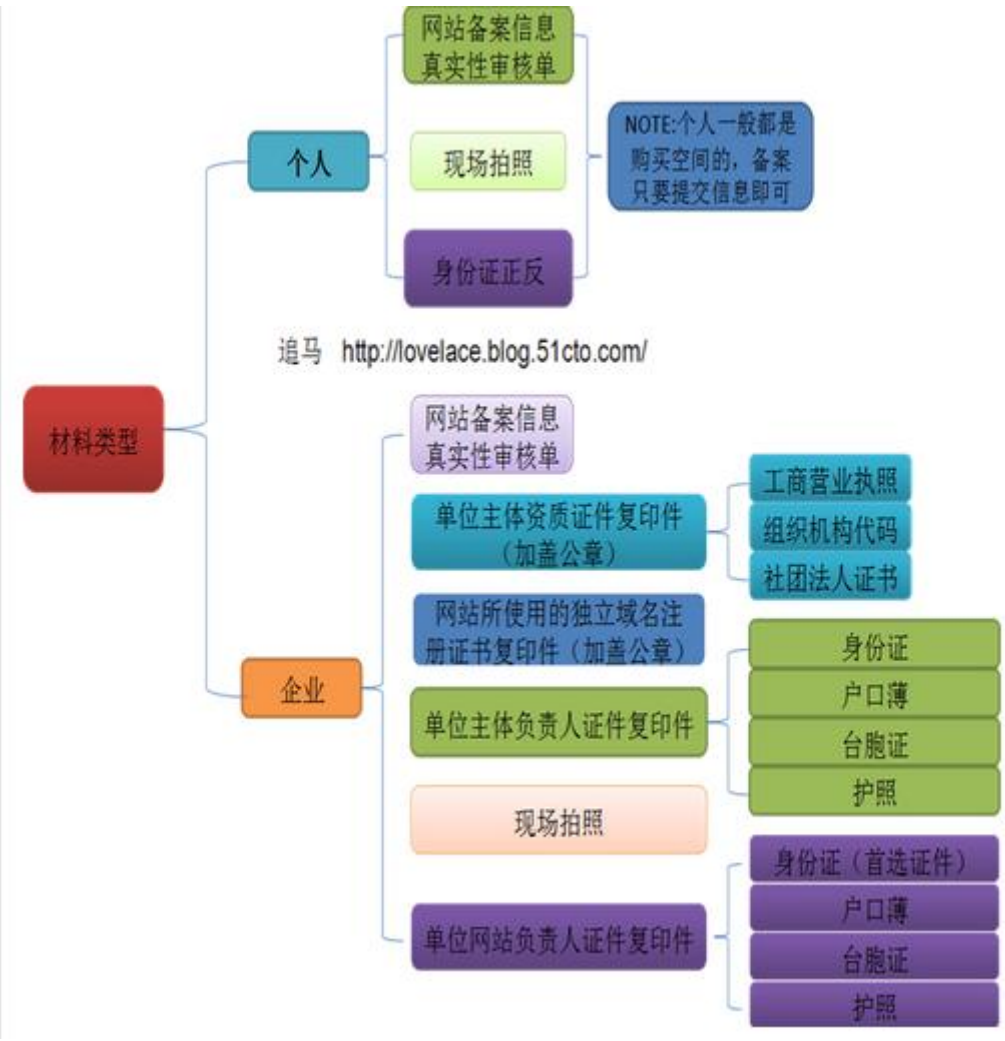
3.8、验证

证书导入完毕之后，可以找一个 whois 网站来进行验证，(这一步在你备案号下来之后就可以检测)，大工告成。

最后用几个图总结下：




企业备案
自备空间
三方外包
相较于公司服务器来讲
材料双份
委托书双份
公司服务器
登陆网站备案系统网站
选择自行备案导航
填写备案信息
注册新账号
选择接入商所在省
添加网站信息
等待接入商电话或邮件通知
接入商名称为你服务器所使用公网ip的供应商
准备材料
现场拍照
下发icp号码
追马 http://lovelace.blog.51cto.com/
等待结果
审核通过
接入商修改信息
购买空间
国内空间商
类似三方外包
相对而言更省心
国外空间商
不做备案
域名到手就用



4、 需要注意的几点

4.1、几种备案方式的优缺点（时间 && 金钱）

个人网站一般都是域名和空间一块购买，备案省时省力。

公司的网站的话一般需要 IT 相关负责人来办理，费时费力。

第三方备案，快速。但是要支出额外费用！200+....

4.2、几种备案方式的安全隐患

个人网站的话需要找稳定的空间商，因为备案之后你要考虑的问题是网站运行是否稳定的。

公司备案的话，这个要进行审核的，如果只是某个人来操作这些事情的话，很可能域名什么的就会随着这个人的离职而产生扯皮情况。

第三方备案的隐患就是说如果你的域名很有价值，那中间你稍不留神就会有猫腻的，具体不再详述，办理的时候注意就行.....

5、个人感受

5.1、使用固定 ip 是哪家的就找家的接入商侧系统去处理

5.2、尽量不要给地区的通管局打电话，你懂得。。。

5.3、如果是企业的域名（而你又不太懂这块），建议还是花点钱办事情

5.4、如果是个人域名的话而且服务器又是自己的，那就慢慢搞。。。

后记：上次备案操作流程近 40 天，当时由于不太懂这块，然后就是网上各种搜索资料，但是官方给出

的流程太过于正规，很多名词不认识，而老板又不愿意多花额外的钱去搞这个（说这个本就是职责范围内要做的事情。。。）然后就各种打电话，各种 QQ 群求帮助，但是收效甚微。。。后来不得已才一边查资料，一边给当地备案中心打电话，我发誓，再也不想拨打那个电话…… 希望写这篇文章对大家理解备案这块有所帮助！

职场观察：高薪需要什么？

作者：孙杰　来源：http://xjsunjie.blog.51cto.com/999372/1378547

新的一年，看到别人跳槽或涨薪，你是否也蠢蠢欲动。怎样拿到高薪？不知你是否想过。勤恳的埋头苦干抑或过硬的技术实力或者出色的沟通能力，你认为是什么起主要决定因素？

在这个变化的年代，我们的方向在哪里？请看 IT 人的成长故事。

王超是我的朋友，来京四年整。最初在一家民企做 LINUX 运维工程师，月薪 5000。工作很认真，埋头苦干型，每天工作时间很长，让加班从来无怨言。即使是周末休假，只要有工作任务也是随叫随到。然而当他提涨薪时，企业说是要考虑考虑。一两个月后这事杳无音讯。他离职了，跳槽到一家私企做系统工程师，月工资 7000，工作稳定，工作内容也固定，继续埋头苦干，每天把自己的工作做好。一年后有涨薪，幅度 10%。这样又两年，工资到了 8500，依然感觉日子很难熬，买不起车更买不起房，羡慕高薪的人。当然他性格内向，不善沟通和与领导交流，技术能力中等，交给他的活也总能干完，但这样一直干下去么，高薪会青睐他么？

李建是我的一个前同事，做 ORACLE 数据库的，OCM 认证通过者，技术能力很强。先是在外企做数据库运维，月薪 12K，因为不喜欢上司的做事风格和公司的管理制度，干了一年后离职。之后跳槽至国内一家比较有名电子商务公司，月薪 16K，处理技术问题很有自己的一套，但是喜欢沉浸在技术世界里，不喜欢与人交流也很傲气。然而技术是越来越厉害了，干了两年还是无法涨薪。每逢遇到问题，总是觉得自己是权威，所有人都得听他的，别人说什么也听不进去，甚至直接顶撞技术总监。而与之同时进公司的一个同事，技术不是很牛但是为人处事不错，很快当部门经理了，工资早已过 20K，心里渐渐不平衡，随后辞职去了另一家私企，还是做数据库运维，工资 17K。高薪依然渐行渐远，他为此郁闷。

赵东我的一个朋友，做项目经理，主要负责云计算和虚拟化的项目。技术好，沟通能力强，有一个 PMP 认证，在京城混了五年多，起初月薪 15K 现在月薪 30K，由于业内人缘不错，马上去一家知名公司，年薪 60 万，算是 IT 界高薪了吧。

纵观 IT 界，高薪有几何？你是埋头苦干型、技术实力派还是善于人际沟通与管理，看完上面三个故事，你有何感想？

关于第一个故事，我的建议和想法是：

现在的人们生活节奏太快，工作也过于辛苦，以至于他们很容易忽略生活与工作的平衡。人们总是自己埋头苦干，很少抬起头来看看到底进展如何。于是，他们便错过了很多改善现状的机会。一个人埋头苦干的时间越长，他就越感到寂寞，而他的工作也越来越容易受到孤立。尽管是由工作引起的，但它的影响还会波及个人生活。生活中除了工作就是工作。而且你做的事情越多，就会有更多的事情要去做。这简直就

是个无底洞啊！你要做的是停下来，花点时间思考，提醒自己，不要再一味地埋头工作，详细地分析一下你工作中存在的问题。为什么这样说呢？因为这就好比你驾驶着汽车以每小时 160 公里的速度在高速公路上飞驰，你考虑的只是保证汽车不偏离车道，而无法注意到路边的风景和留心你身边的人在做什么想什么。当你静下心来，开始思考工作以外的一些事情，关注自己身边的人和事，你就会发现自己的问题。当你找到了问题，继而明确下一步前行的方向，你的涨薪才有希望！

关于第二个故事，我的建议和想法是：

技术也很重要，但最重要的是做人的风格，学会与同事和领导相处。现在很多 IT 人除了技术什么都不懂，整天得罪人。琐碎的小事不愿做，关键的大事搞不定。哪怕你是 CCIE、RHCA、OCM 全考出的，我不用你难道地球就不转了？再说了，你考的高级认证越多，无形中别人对你的期望值也越高，你实际的技术能力是否和那些高级认证匹配，这些在企业里都是需要考量的，几张证书不能说明什么。切记，技术是用来解决问题的，不是拿来炫耀和自傲的，更何况很多技术在企业也不是全部用得到的。先做人后做事，古有此理也。心态转变一下，世界可以变的更美好，高薪也不是遥不可及。

关于第三个故事，我想大家可能跟我的想法一样，选择前沿的技术，把准方向，有技术有人缘，好的工作机会总是等着你。关于以上，如果你有更好的想法和建议，欢迎交流。

总结一下，IT 界不是没有高薪，关键是看你自己的能力。对于还没毕业的同学，我希望你们能先认真读书，至少拿个学士出来，同时学好英语，多参加社会活动，即使你作 IT，技术也不过只有 40%的比重而已，重要的是沟通和为人处世的技巧。对于职场人士，引用老男孩的一句话“技术是根基，沟通是桥梁，思想是灵魂。” 因此，光会技术是远远不够的，这点大家一定要认识清楚。

当然，职场中还有很多现状是不合理的，你也看到某人不学无书也没什么能力依然拿着高薪，或者只因他的机遇好很容易的拿到了高薪，但这只是个别现象，我们不能以偏概全。很多时候，我们仍然要靠自己的实力和不懈的努力，才能走向更大的舞台，实现自己的高薪梦和人生的价值。现实中有些东西，如果你不能改变它，那要么你适应它，要么你毁灭。在沙漠里谁能活下来？是万物之长的人还是骆驼？ 所以物竞天择适者生存，为了高薪，我们需要做的是不断改变和提升自己。

IT 人的自我导向型学习：学习的 4 个层次

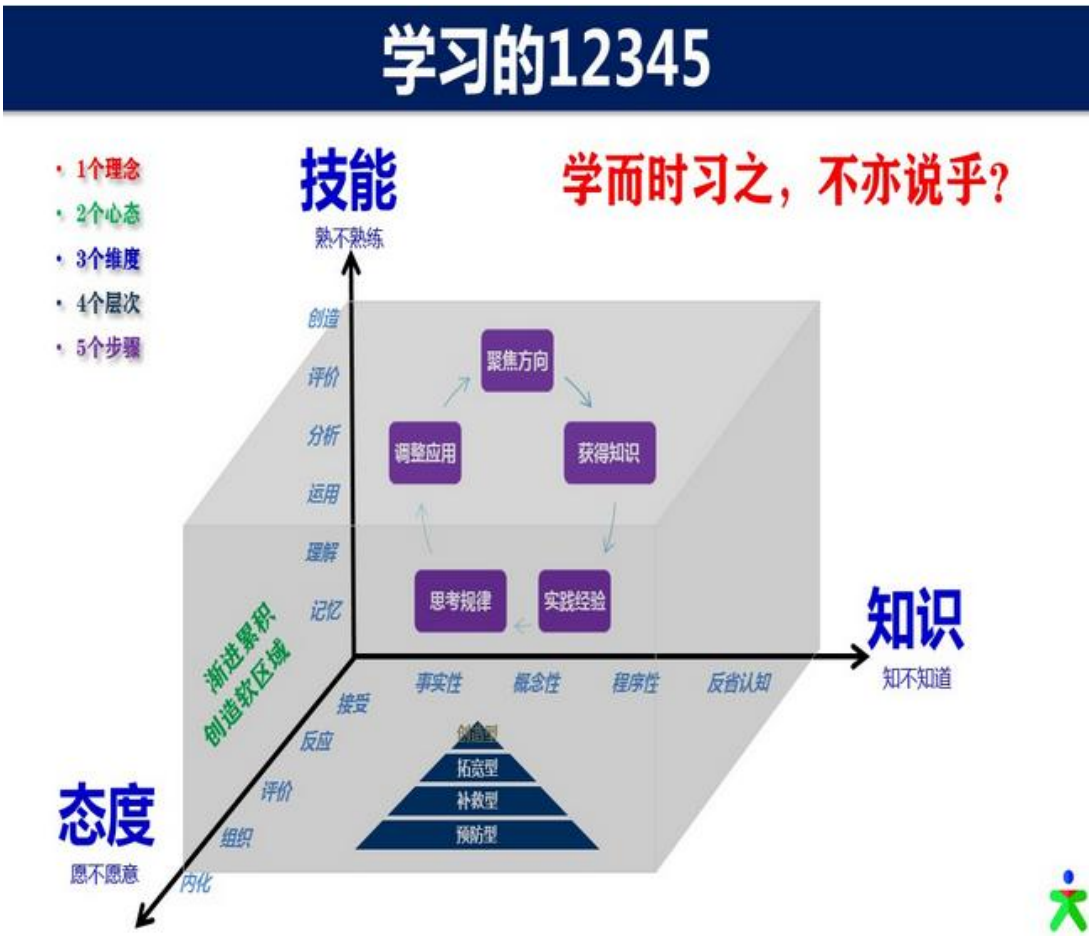
作者： 周金根 来源：http://zhoujg.blog.51cto.com/1281471/1382004

谈起软件开发一定会想到用什么技术、采用什么框架，然而在盛行的敏捷之下，人的问题逐渐凸显出来。不少企业请人来培训敏捷开发技术，却发现并不能真正运用起来，其中一个主要原因就是大家还没有很好的学习能力。没有学习，就不会有合格的 ScrumMaster，没有懂得敏捷本质的成员，没有 Being 敏捷的思想。其实学习敏捷开发本身就是一种学习，敏捷实践中也都是学习，学习无处不在。学习那么重要，但又有多少人不仅是爱学习，而且有学习方法呢？闲话不多说了，继续和大家侃侃 IT 人的自我导向型学习，这可是敏捷个人体系三个组件之首哦。

学习的 3 个维度

按照惯例，大家闭眼想想敏捷个人的学习 3 个维度是什么？.....

上一篇我们讲了敏捷个人时中法的自我导向型学习的 3 个维度：知识、技能、态度，关于这个话题可以延伸出很多话题，例如敏捷个人的头手心模型、高效执行力等。其实之前也分享过一些内容，例如 学习 = 知识 + 实践 + 思考 + 心按态 之类的，只是那些都是零碎着写的，还不能成体系。这次写这个系列呢，一方面是分享，另一方面是在分享的过程中整理成一个个人学习的有价值的参考方法，所以大家在看的过程中有什么可以分享给大家的也望大家在回复中不吝赐教。



如果你忙，你在学什么吗？
开始之前，你回答我一个问题，“你在学习哪些东西？”

……

再回答我下一个问题，“这些东西对你有什么用？”

*&%……&%&……@

我知道，有很多人并不知道自己在学什么，就是知道自己很忙，不停地在学，你们有上进心，知道不能让自己闲下来，有点事情干表明自己也有追求嘛。这也不能怪你，学校告诉你如何学习是告诉你如何学课本考大学；企业招你是让你挣钱，但是也没有太好的方法指导你学习，除非你遇到一个好师傅，不过大部分人难以遇到。这其中许多人不认为自己学习需要什么方法，完全按照自由意志，随性而学，想到英语重要就开始背单词，坚持不了一两个月就取消了；看到 ruby 开始在身边流行起来了，就想去学习一下，看了一两本书什么都没做就草草了事；想着学谈吉他，报了个班，没多久也不去上课，吉他也开始布满灰尘。你可能一直在努力学习，不过因为不是有意识的去做并做好，所以你很容易半途而费，或者效率上大打折扣。

你是忙碌的人吗？如果是，如果有敏捷个人帮助你，告诉你该如何学习，你愿意听吗？

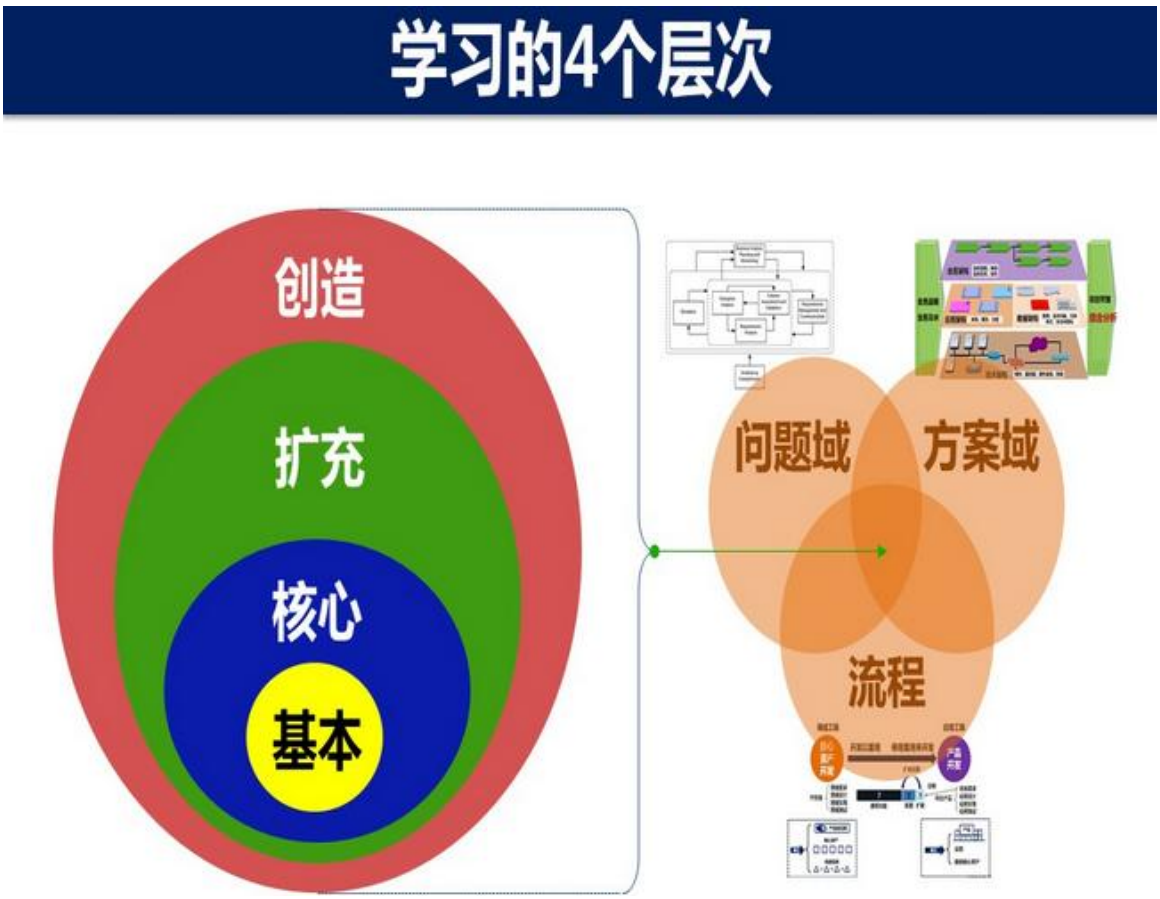
如果你闲，你在玩什么？

我知道，其实现在闲的无聊的你，其实一开始属于有追求，忙碌学习的人，只是耐不过时间的磨练而把你推倒这个群体了。谁不想出人头地，赚大钱，谁不曾努力过，不过有用吗？领导不赏识，同事不配合，环境不满意，家人不支持，自己运气不好，一串串障碍把你逼得没办法才开始放弃自己。如果真是这样，我先表示一下同情，你真的“成熟”了。不过即使你真的很闲，也没什么，很多人和你一样，一年觉得无聊一次，一次半年到一年。幸好，这只是过去的你，将来的你还没出现；幸好，你开始看如何学习、如何成长的非技术性文章；幸好，你还有一颗不死的心。从小到大就一直在学习，你也应该明白一件事情，从毕业工作开始，就没有人总在耳边逼你学习了，因为以前逼你学习的改成逼你结婚、逼你挣钱、关心你活的好不好了。

小时候可能会说，读书是为了父母的面子。但是如果我们现在还天真的以为学习是为了父母，那你真是还没长大。我们也不小了，现在也应该明白了，学习本来就是一件应该自觉的事情，特别到了成年人，它的主观色彩非常浓，再靠人灌输是很难学到真本领的。希望无聊的你开始慢慢在心里重新树立一种信念：为自己而学，为了自己的价值而学习，为了自己的快乐而学习。

4 个层次模型图

每篇文字不给大家一张我的 PPT 总觉得少了什么，今天给大家奉上学习的 4 个层次模型：



我们先来看看左边的 4 个层次，分别是：基本知识、核心知识、扩充知识和创造知识。
基本知识是你工作必须掌握的，否则你连养活自己的能力都不够，这相当于初中级水平；
核心知识是在工作中你负责的那部分工作所需要掌握的，这相当于高级和纯专家级水平；
扩充型学习是通过从横向或纵向领域扩充中拓宽自己的视野和知识，这相当于多面手专家；
创造型学习是让自己提高到另一领域，在这种学习中，学习者已经不满足于获取别人堆在他们面前的知识，他会在掌握某种新知识之后创造新的知识，这相当于大师级领袖。

你十年后想达到哪种学习水平呢？今年准备达到哪种学习水平呢？先想一下这两个问题再看下面，最好把这两个答案在本文后回复一下。

学无止境，人生就是一个不断学习的过程，从每个人的职业发展来看，基本可以归为三个阶段：生存期、发展期和实现期。刚工作的人处于生存期，最重要的事情就是学习基本知识来赶快适应工作，否则可能不被录用。但是仅仅被录用并不长久，除非你的公司肯留下碌碌无为的人，或者你能接受碌碌无为的自己，否则你一定会开始关注自己的核心知识，以便将来可以凭借自己所擅长的技能更好的养活自己还有家人。其实到这一步，成为了高级人才或专家，基本上就可以不用为钱发愁了，不过这时你会开始做更多事来体现自己的价值，你会希望成为多面手，不过好在你是专家型的多面手，懂很多，但还有一两门自己的专长。做到最后，你会开始思考自己了，为了更好的实现自己，你开始有了自己的体系，这就是开始再创造知识了。

说起来好像是这么回事吧？好像谁都知道，但为什么有的人就是掌握不了这么多知识呢？是笨吗？可以这么说吧，但不是智商笨，而是不懂得该在哪里发力，也就是不知道自己学习的方向在哪里。

你有学习方向吗？

你在回答上面提出的希望今年想在哪个层次水平学习了吗？在你回答的时候，你确定了是学习什么了吗？如果没有，那你的回答是没有用的。为什么这么说？因为学得愈多会发现要学的就更多，你不可能所有的方向都做到专家级，如果你回答都是大师级、专家级的话，那么现在你再好好想想，你想学习哪些东西来做到这个级别？

关于学什么这个问题，我们来想想你为什么要学习？你说你学习是为了消遣，那你不在本文读者之列；如果你能说出你为什么学习，答案一定和你要做的任务或目标有关，也就是解决一个问题。而现在有时候一件事情不是一个人能搞定的，于是开始分工协作，但是完成事情的步骤却不会少，知识分工不同而已。简单的说，完成一件事需要掌握的知识有三块：问题域、方案域、流程，这也是我在学习模型右半边画的学习内容的 3 个主题域。问题域和方案域是我借用我在研究模型驱动开发中的术语，用来帮助明确要解决什么问题以及有什么方法、工具和方案来解决或改善，而流程你也能明白，就是用来保证实施方案来解决问题。这三个领域的学习其实各自有自己的一些知识要点，对人的思维方式要求也不一样，这时我们就要问自己了，你到底喜欢发现问题还是解决问题，或者辅助实施呢？还是都喜欢？不管你是喜欢一个，还是什么都不喜欢，或者什么都喜欢，你一定要选择一个作为你的核心方向去学习，成为这个领域的专家。

如果你够顺利的话，你可以继续扩充和创造你已经掌握的核心知识，如果你想成为多面手的话，还可以开始考虑把你的精力分一部分在其他领域。 你不用去百度或 google 搜索以上分类理论了，以上只是我基于自己从事工作多年的经验对个人学习的知识内容的分类，不一定很严谨，不过它一直在指导我自己学习。我们做软件开发，其实不就是一直围绕着这个在转吗？模型驱动开发与问题和方案域、BABOK 与问题域，TOGAF 与方案域、软件产品线与流程，这些虽不是完全匹配，但是也都各自紧密相连，这也是我选择学习内容背后的思考。

也许你可能希望我在这里列举出对于刚工作的人、工作 5-6 年、工作 10 年以上等分别要学哪些知识？这个我真不敢在这里列举出来，因为我知道，每个工作具体要求都不一样，每个人兴趣和能力都不一样，知识技能型的学习内容自然也哦度不一样，不过我倒是可以给大家列举一下不管什么阶段都要学习的自我反省型知识内容，那就是要学习自控力、学会自律、学会管理自己的情绪、学会如何与各种人打交道、学会沟通和团队协作，这也是我学习敏捷个人的自我导向型学习、目标时中法管理和创造幸福生活三大系统的原因。

我的周围有一部分能够找到自己的核心方向，有部分人却还在迷茫不知朝哪里发力，还有一部分嗯？那就是方向太多，也就是兴趣太多。没有兴趣是一件可怕的事情，兴趣太多也是一件头痛的事情。

有人问过我一个问题，她说很困惑自己兴趣太多，以至于自己很疲惫和什么事都没干成。兴趣多了方向也多，方向多了其实就是没有方向，那我们应该如何看待兴趣以及如何解决与学习方向的冲突呢？

对于一件事物，我们首先要去认识它，这在敏捷个人 101 每日发现练习中有很多类似的话题。对于兴趣，我认为你的每种兴趣都值得体验和尝试，因为兴趣只存在于人的内心。当兴趣消失时，你才应该抛弃

它。你也许不知道自己的兴趣能否坚持到底，但是兴趣能将你带到哪里，谁也不好说，所以绝不要仅因为觉得自己不能坚持这一点而放弃对某种事物的学习。 但是在兴趣学习中一定还要认识到，学得多不是你的目标，给每种兴趣一个公平尝试的机会，但我们一定要一次只专心做一件事情。按照我前面所说的学习层级，当你遇到一件感兴趣的事情，那就把它学到接近核心知识层级，如果你没有学到，那么你就立刻干掉这个兴趣吧，这不是你的兴趣，而是头脑发热而已。如果你学到了核心层级，但是在掌握了一定的基础之后发现没有兴趣了，那也干掉。

不要问我你应该培养什么兴趣？知道自己有没有兴趣比知道有没有能力更简单，这种感觉本身以及取得进展的迫切要求，往往不言而喻。你该学什么？学什么要由你自己决定，这取决于你最需要什么，最喜欢什么，以及怎样做才能给你带来财富、提高地位以及所有与你相关的东西。

你真的没有优势吗？

你也许说按照我说的，你也选不出来应该学什么，因为你不了解自己，特别是还不确定是否某个方向真的适合自己，或者自己努力真的能成为专家。

你了解自己的能力吗？你完全清楚自己能做什么，不能做什么吗？如果答案肯定，那你真是厉害，可谓百里挑一，这种自知之明的人可是世间少有。其实大多数人自称知道自己能做什么的时候，也只不过是曾经接受过良好的训练而已，而社会在变，事情本身也在变，这种肯定的回答其实也是一种运气。

既然没有人知道自己能否做成某件事，那就看谁有天赋。这就更扯了，有多少人是真的一看就具备某种天赋的超人呢？我们大部分人都是天生都不知道很多事情，然后靠着自己的努力和学习才获得认识，我们可以相信相信别人具有某方面的天赋，但是一定不要低估自己后天学习带来优势。

也许你曾经努力过学习，不过失败过一两次，于是决定自己真的不行。其实，学习本身是可以学习的，千万不要因为一时的学习停滞就否定自己的学习能力。学习本来就是有一个获得曲线的，这个在下篇分享中会说。在识字、行走、睡觉、用筷子吃饭、穿衣中，哪些行为不是通过你学习获得的？你学走路花了多长时间？

不要指望学习所有知识速度都一样快，也不要指望学习某门新学科从头到尾都能保持统一速度。一般在刚开始时，我们大量阅读相关书籍和动手做简单练习后就可以快速进步，例如我今天学了一天 Unity3D 就可以做一个简单的 3D 游戏了。但是，我们要认识到，紧接着你便要经历进展缓慢的阶段了，除非你跨界学习或者换了一种思维方式给你带来了新的大跨度学习，否则还会成长，只是没有想象那么快而已。

许多抱着学习心态的人进入第一个进展甚微甚至毫无进展的漫长停滞期时便会显得灰心，对自己失望。我觉得你大可不必这样，前面我已经说了这是必然，你这样的负面情绪只会造成你对学习的不满，你应该接受这种速度延缓现象是正常的，它也是你消化知识的一个必经阶段。

总的来说，一种实际操作越是复杂、掌握的速度就越慢，就像一次能学会滑雪，但是要想在没有人帮助和指导下学会 TOGAF 并开始应用却很慢。当你脑子转不过来是，还不如放下学习休息一下，说不定还有

突发的灵感呢。

勇敢的给自己制定学习方向

在学习一门新学科时，要尽早给自己制定一项艰巨任务，这个任务即便不需要数年时间，也需要好几个月的深入研究和思考方可完成。不要让这项任务成为你的负担，而要成为你所想象的最有意思的冒险，将它作为你你职业生涯的一个完整组成部分，把时间和精力用上。

你现在会学习了吗？如果你能够承认自己之前不会学习，这已经是一个不小的进步了，这说明你是个有勇气的人。反正都有勇气了，那就干脆再勇敢点，按照下面敏捷个人学习规划简易版表格来给完成自己的一个学习规划，并在后面回复中给大家分享一下：

我想完成的事情是_____，这件事情开始时间是_____ ，结束时间是_____，我要先研究一下内容_____来搞清楚自己是不是对此感兴趣。在学习第一阶段，要解决的主要问题是_____，我已经掌握于此联系紧密的知识有_____，为了实现这一目标，我每周要留出____小时来学习_____提高自己的能力。
我是否有认识某个能给我提供信息的熟人_____
最好学习的书目有_____
该领域最主要的权威是_____

也许工作中你做过太多应付差事的规划了，你也已经讨厌其做规划的事情了，没想到想放松一下来到这里又看到这种莫名的建议。上面的自我学习规划问题虽然比较简单，不过确实比较有效，不信你敢试着问问自己吗？如果你没想出来，那可能就会无意识的在学习，效率就会大打折扣了。

这次内容比上次的好懂些，不过内容不少啊，学习的 5 个步骤下次再说，如果喜欢的话，不要忘记继续给我留言点击推荐啊，这次超过 30 人赞继续下一篇啊：)

本次练习你要做的
你十年后想达到哪种学习水平呢？今年准备达到哪种学习水平呢？

按照敏捷个人学习规划建议版表格完成自己的一个学习规划

向身边的人介绍已经学到的敏捷个人自我导向型学习

扫描二维码，加 51CTO 博客为朋友

关注 51CTO 博客微信，打造你的个性！

http://blog.51cto.com

有奖调查：你坚持写博的动力是？

编后语

《51CTO 博客月刊》为 51CTO 博客整理出品

如果您有意投稿，请联系我们
如果您有反馈意见，请告诉我们

联系方式：
Email：blog@51cto.com
QQ: 1173854158

欢迎关注我们：
@51CTO 技术博客